CENA 10.000 zł

ISSN 0867-3918 INDEKS 377112

64 PLUS 4 & AMIGA

4/'92

MIESIĘCZNIK UŻYTKOWNIKÓW KOMPUTERÓW COMMODORE



Drodzy Czytelnicy!

życzymy Wam zdrowych, pogodnych i słonecznych Świąt!

Redakcja.

OD REDAKCJI

Informujemy, że nasze pismo można w dalszym ciągu **zaprenumerować** - co daje pewność systematycznego otrzymywania (drogą pocztową). Nasz miesięcznik kosztuje w prenumeracie 10.000 zł. Prenumeratę można zawrzeć na okres nie krótszy niż dwa miesiące, w dowolnym okresie, maksymalnie do końca roku kalendarzowego. Wykupujący prenumeratę nie ponoszą kosztów przesyłki pocztowej.

Wadliwa dystrybucja „64 plus 4 & Amiga" przez przedsiębiorstwo RUCH jest przyczyną kłopotów, jakie mają czytelnicy chcący nabyć nasze pismo. Zdarza się, że otrzymujemy jako zwroty NIETKNIĘTE paczki zbiorcze. Tymczasem czytelnicy sygnalizują, że do wielu kiosków nie dociera ono wcale. Dlatego:

zapraszamy wszystkich chętnych do prowadzenia kolportażu „64 plus 4 & Amiga" (kluby, studia i sklepy komputerowe, księgarnie, osoby indywidualne itd.) do współpracy!

Oferujemy korzystne warunki!

Zainteresowanych prosimy o kontakt z działem dystrybucji pod adresem: Przedsiębiorstwo ABUK, 87-200 Wąbrzeźno, ul. 1 Maja 33.

Przedsiębiorstwo ABUK S-ka z o.o. oferuje państwu **szybką i tanią obsługę reklamową.** Ogłoszenia drobne od osób indywidualnych (do 10 słów) przyjmujemy bezpłatnie. Większe - 1000 zł za słowo. Reklamy ramkowe (minimalny format - 20 cm^2): 1cm^2 ogłoszenia **-8000zł, cała strona - 3,0 mln zł**; każdy kolor - odpowiednio 100% drożej. Ogłoszenia przyjmujemy za pośrednictwem poczty (nasz adres - patrz stopka redakcyjna). Treść ogłoszenia z określeniem formatu reklamy (ewentualnie zamówieniem koloru) prosimy nadsyłać listem poleconym wraz z odcinkiem wpłaty. Wpłat prosimy dokonywać za pomocą przekazu pieniężnego na konto Przedsiębiorstwa ABUK, Bank Polska Kasa Opieki SA Oddział w Bydgoszczy, konto nr : 5.09011-400522.7-2511-30-111.0. Dołączenie do zamówienia odcinka wpłaty przyspieszy zamieszczenie reklamy. Redakcja nie ponosi odpowiedzialności za treść i wiarygodność ogłoszeń.



miesięcznik nr 4(18)
kwiecień 1992
cena 1 egz.: 10.000

64 PLUS 4

WYDAWCA: ABUK Spółka z o.o.
REDAGUJĄ: Waldemar Szczygieł (redaktor naczelny) z zespołem.
ADRES REDAKCJI: Redakcja „64 plus 4", 85-166 Bydgoszcz 43, skrytka pocztowa 64.
OKŁADKA: Piotr Bartz.
SKŁAD: ABUK
DRUK: W.Z.G. Wąbrzeźno, okładka: Z.P. POLRASTER, Bydgoszcz.

W numerze :

Od redakcji 3
Z daleka i z bliska 4
23 klawisze funkcyjne dla C-16, +4 5
Niezniszczalni 5
Sample mon v.20 7
Sprite'y i Basic 8
Spis zestawu PDP na C-64 (nr 15) 10
Modemowanie 11
Assembler 6510 - lekcja 9 13
Public Domain Pack . .15
Reklama 17
Party w stolicy 19
Utopia 20
Mega-Lo-Mania 21
Pakujemy chłopaki . . .22
PDP na Amigę zestaw nr 15 25
Kącik początkującego kodera 26
Odpowiedzi na listy . .30

NOWINKI



Na tegorocznych targach CEBIT'92 firma Commodore zaprezentowała kilka nowych produktów. Oto ich krótkie opisy.

- Najmłodszym dzieckiem firmy Commodore jest nowy model Amigi o numerze 600. Amiga 600 to zupełnie zmieniona wersja 500-tki. Nowa Amiga nie posiada portu rozszerzenia, co uniemożliwia podłączenie niektórych urządzeń zewnętrznych (twardy dysk, moduły typu Action Replay czy Power-X). Są jednak dostępne modele standardowo wyposażone w twardy dysk o pojemności od 20 do 120MB. Amiga 600, podobnie jak poprzednie modele tej serii, jest wyposażona w szesnastobitowy procesor Motorola 68000, pracujący z częstotliwością zegara 7.14MHz. Pamięć tej Amigi można maksymalnie rozszerzyć do 10MB. Model ten ma również wbudowany modulator. Zastosowanie nowych układów Agnus i Denise umożliwia wykorzystanie nowych trybów pracy: Super Hires, tryb Productivity oraz tryb A 2024. Pierwszy z nich oferuje rozdzielczość 1280x256 w trybie interlace. Productivity umożliwia uzyskanie rozdzielczości 640x480 (non interlace) i 640x960 (interlace) przy 16 kolorach. Tryb ten jest szczególnie polecany przy pracy z programami obróbki tekstów, kalkulacyjnych arkuszach i DTP.
  W trybie 2024 do dyspozycji mamy rozdzielczość 1008x1024 (programy CAD-CAM, DTP).

- Drugim prezentowanym na targach modelem była Amiga CDTV. Jest ona bardzo podobna do odtwarzacza płyt kompaktowych. Do Amigi CDTV można podłączyć dodatkową stację, klawiaturę, monitor oraz mysz, co przekształca ją w zwykłą AMIGE 500. Można także podłączyć twardy dysk o pojemności od 20 do 120MB. Amiga CDTV ma możliwość odtwarzania muzyki z płyt kompaktowych używając jej np. jako podkładu do programów. Stacja dysków optycznych Amigi CDTV jest zgodna ze standartem ISO-9660 i ma pojemnoć 550MB.



- W końcu Commodore przedstawił przystawkę do Amigi 500 zamieniającą ją w Amigę CDTV. Wadą jej jest brak możliwości podłączenia standardowego dysku twardego.

- Zaprezentowano również nową kartę do Amigi 2000/3000 emulującą komputery PC 80386. Karta posiada procesor Intel 80386SX taktowany częstotliwocią 20MHz.

- Pokazano również nową, wewnętrzną stację dla Amigi 2000/3000 umożliwiającą zapis 1.76MB na dyskach 3 1/2" HD.

- Commodore przedstawił także swoje modele zgodne z IBM PC. Są to:
  -komputer z procesorem Intel 80486DX w jednej z trzech typów obudów: desktop, wieża oraz miniwieża,
  - z procesorem Intel 80486SX taktowanym zegarem 20/25MHz,
  - nowe modele z procesorem Intel 80386 oraz Intel 80386SX, a także Intel 80286.

- **W tym roku Commodore ma zamiar zorganizować targi komputerowe pod hasłem:**

  ŚWIAT COMMODORE - AMIGA '92 - NOWY WYMIAR!!

  Targi odbędą się w Frankfurcie nad Menem w dniach 26 do 29 listopada br.
  Na powierzchni 20.000m2 (w hali nr 5 i 6) firma zaprezentuje swój „cudowny świat komputerów". Szczególnie ciekawie zapowiada się dział „Pro - Amiga", w którym prezentowane będą min. nowa Amiga 600/600HD.
  Na targach będzie można dokonać zakupów sprzętu. Poprzez specjalnie zainstalowany telefon każdy będzie mógł porozumieć się z sztabem ekspertów, którzy będą odpowiadać na wszelkie pytania.

23 KLAWISZE FUNKCYJNE DLA C16, +4

C-16

Korzystając z poniższego programu można przyporządkować 23 instrukcje w Basic'u odpowiednim klawiszom - programowanie lub wpisywanie listingów będzie dużo łatwiejsze.

Wywołanie poszczególnych funkcji uzyskuje się przez naciśnięcie klawisza ze znakiem angielskiego funta, a następnie klawiszy od A do W (wywołanie odpowiedniej funkcji). Jeśli przypadkowo naciśniemy ten klawisz, to naciśnięcie X spowoduje „wycofanie się".

Listę rozkazów Basic'a przyporządkowanych odpowiednim klawiszom znajdziecie w liniach DATA - 1130 do 1170. Własne przyporządkowanie funkcji możecie uzyskać przez zmianę treści tych linii. Jeśli chcecie - przykładowo - wywołać funkcję LOAD zamiast LIST klawiszami <$ zmieńcie w linii 1150 wyraz LIST na LOAD.

Należy zwrócić uwagę, aby występujące funkcje nie były zbyt długie, w przeciwnym razie komputer wyświetli komunikat ILLEGAL ERROR IN 100.

Jeszcze drobna uwaga. Nasza procedura (o długości 80 bajtów) zajmuje bufor magnetofonu i RS232. Z tego powodu nie można w czasie jej wywoływania przeprowadzać operacji z magnetofonem, gdyż spowoduje to skasowanie linii rozkazów.

```
20  A=1056
30  FOR I=0 TO 9: FOR U=1 TO 8: READ X$:
    X=DEC(X$): P=P+X: POKE A,X: A=A+1: NEXT U
40  READ P1$: P1=DEC(P1$)
50  IF P1P THEN PRINT "BLAD DANYCH W LINII";
    I*10+1000: END
60  P=0: NEXT I
70  :
80  AD=833
90  FOR I=0 TO 23
100 POKE 1032+I, AD—768: READ X$
110 FOR A=1 TO LEN(X$): X=ASC(MID$(X$,A,1):
    POKE AD,X: AD=AD+1
120 NEXT A
130 POKE AD,0: AD=AD+1
140 NEXT I
150 :
160 PRINT "{CLR}___{RVSON})(RVOFF}{
    RVSON}{RVOFF}D{RVSON}M{RVOFF}RC{
    RVSON}L{RVOFF}SC___C{RVSON}@": SYS 3072
170 END
180 :

1000  DATA A5,D8,D0,0B,A5,C6,C9,02,048E
1010  DATA D0,02,85,D8,4C,42,CE,A5,0430
1020  DATA EF,48,20,9F,FF,A4,EF,B9,0541
1030  DATA 26,05,C9,41,90,29,C9,59,0310
1040  DATA B0,25,A8,68,85,EF,A9,00,0402
1050  DATA 85,D8,98,48,20,4F,FF,9D,0448
1060  DATA 01,00,68,A8,B9,C7,03,A0,0334
1070  DATA 03,20,88,90,E6,EF,A4,EF,04A3
1080  DATA A9,01,99,26,05,D0,03,68,02A9
1090  DATA 85,EF,10,C0,00,00,00,00,0244
1100  :
1110  REM *** LISTA ROZKAZOW ***
1120  :
1130  DATA AUTO,CLOSE,COLOR,DELETE,END
1140  DATA FOR,GOTO,HEADER,INPUT,JOY
1150  DATA KEY,LOAD,MONITOR,NEXT,OPEN
1160  DATA PEEK,POKE,RENUMBER,SAVE,TRON
1170  DATA TROFF,VERIFY,WAIT,"{LEFT}"
```

Na podst. 64'ER 10/88/46

NIEZNISZCZALNI

Pan Andrzej T. Witbort nadesłał do nas list, w którym przytacza min. szereg ciekawych „poprawek" do gier. Mamy nadzieję, że temat ten zainteresuje szersze grono użytkowników C-16 i +4.

/.../ Ze względu na wbudowany MONITOR języka maszynowego, dokonanie poprawek nie nastręcza zbyt wielu trudności - przerywa się program przez jednoczesne naciśnięcie klawisza RUN/STOP i RESET. Po dokonaniu poprawki wykonać należy rozkaz G... (adres początku programu). Trudności mogą wystąpić jedynie w przypadku, gdy zmieniony został wektor przerwania RESET (nie można przerwać programu), lub gdy program usuwa, w trakcie pracy, procedurę inicjalizującą (nie można programu uchronić drugi raz).

W przypadku bardziej skomplikowanych poprawek podam szczegółowy opis czynności.

Zacznijmy od prostych poprawek:

BIG MAC	$ 31A16	LDA #$ XX	-ilość „żyć"
	G 1B58		-start programu
BOMBER	$3A80	LDA #$ XX	-ilość „żyć"
	G 3A00		-start
COMMANDO	$ 1654	LDX #$ XX	-ilość „żyć"
	G 2677		-start

C-16

EXORCIST $ 3F55 SBC #$01 -odejmuje po jednym punkcie energię
G 11E0 -start

FINGERS MALONE
$ 27CA SBC #$00 -nie odejmuje
$ 1888 ADC #$000 „żyć"
$ 17CE ADC #$00
G 16A0 -start

FIRE ANT $ 1B56 LDA #$ XX -ilość „żyć"
G 3F90 -start

GHOST'N'GOBLINS
$ 1028 LDA #$ XX -ilość „żyć"
G 101B -start

JETBRIX $ 22A0 NOP -nie odejmuje „żyć"
$ 22A1 NOP
$ 22A2 NOP
G 3000

JET SET WILLY 2 $ 10B5 NOP -nie odejmuje
$ 10B6 NOP „żyć"
$ 10B7 NOP
G 1000 -start

MAGICIANS COURSE
$ 219D NOP -zatrzymanie upływu czasu
$ 219E NOP
G 4700 -start

3D MAZE $ 3038 LDA #$ XX -ilość „żyć"
$ 30A2 NOP -nie odejmuje
$ 30A3 NOP „żyć"
$ 30A4 NOP
G 301D -start

MAJOR BLINK
$1176 LDA #$ XX -ilość „żyć"
G 108A -start

OUT ON A LIMB
$ 13C7 LDA #$ XX -ilość „żyć"
G 0FB0 -start

P.O.D. $ 24FB NOP -nie odejmuje „żyć"
$ 24FC NOP
$ 24FD NOP
G 1F00 -start

SPIKY HAROLD
$ 2231 RTS -nie odejmuje „żyć"
G 1000 -start

VIDEO MEANIES
$ 1ABD JMP $ 1AC6 -unieruchomienie wrogów
G 2A10 -start

GWNN $ 36A8 BEQ 36BF -nie odejmuje
$ 1110 RTS energii
G 3800

Teraz pora na poprawki, których wprowadzenie jest bardziej skomplikowane.

Aby przerwać program, jeszcze zanim zostanie on uruchomiony, należy:

1. Jeśli program jest napisany w systemie TURBO S/L:

a - w momencie, gdy uruchomiony został program TURBO (na ekranie zaczynają migotać kolorowe paski) należy wcisnąć klawisz RUN/STOP i trzymać go aż do końca ładowania programu (ponieważ programy napisane w tym systemie uruchomiane są przez instrukcję w BASIC'u, można przerwać je w ten sposób);

b - instrukcją MONITOR należy przejść do trybu pracy monitora i dokonać poprawki;

c - przez X należy powrócić do BASIC'a i wykonać RUN aby uruchomić program.

2. Jeśli program jest napisany w systemie NOVACOPY:

a - w momencie, gdy uruchomiony został program TURBO (na ekranie zaczynają migotać kolorowe paski) należy wyłączyć magnetofon i nacisnąć jednocześnie klawisze: RUN/STOP i RESET;

b - w bloku programu TURBO (od adresu $ 0140) należy odnaleźć adres startu programu (przeważnie znajduje się pod adresem $ 0196) i zamienić go na $ F44C. Następnie należy odrobinę cofnąć taśmę, wykonać G 0140 i wcisnąć PLAY w magnetofonie;

c - po skończeniu ładowania można dokonać poprawek i uruchomić program wykonując G ...(adres, który poprzednio znajdował się w bloku TURBO).

Są to dwa przykładowe sposoby przerywania programu przed jego uruchomieniem, po załadowaniu kasety. Niezbędne jest to w przypadku gry (posiadam wersję w TURBO S/L):

DEMOLITION - po załadowaniu programu, w linii $ 10FA (w której znajduje się rozkaz JMP $ 5000) wpisać ciąg JMP $ F44C, przejść do BASIC'a i wykonać RUN. Nastąpi dekompresja programu, po której możliwe będzie wpisanie poprawek:

$ 26B2 NOP -nie odejmuje „żyć"
$ 26B3 NOP
$ 117D LDA #$ XX -poziom, od którego zaczyna się gra (od $ 00 do $ 23)
$ 2DD7 NOP -nie przerzuca do poprzedniego poziomu z komnat „z kotem"
$ 2DD8 NOP
G 5000 -start programu

A.C.E - poprawienie tej gry jest jeszcze bardziej skomplikowane, gdyż istotna część programu ładowana jest do pamięci kolorów i ekranu.

Posiadacze komputerów z 16KB muszą znaleźć obszar pamięci, który nie zostanie skasowany podczas ładowania programu, posiadaczy 64KB problem ten nie dotyczy - obszar wolny zaczyna się dla nich od $ 4000.

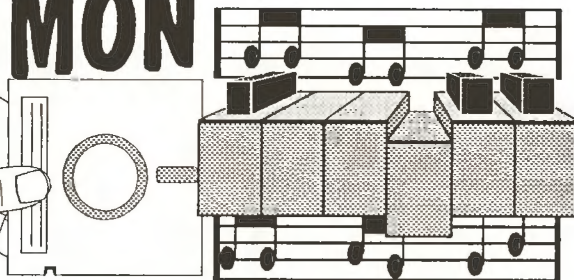
W obszarze tym należy, przed załadowaniem programu, umieścić krótką procedurę:

```
ORG.:  LDA #$ B0
       STA $ 2FBB
       LDA #$ 00
       STA 4 2FBC
       JMP $ 3D3E -adres początku programu
```

Adres początku programu A.C.E. należy ustawić na adres naszej procedury (ORG) i załadować program do końca. Dzięki poprawce, wszystkie rakiety w naszym myśliwcu będą automatycznie kierowały się na cel.

Andrzej T. Witbbrot

SAMPLE MON V2.0



C-64

Ostatnio sampling na C64 przeżywa drugą młodość. Kiedyś wiele grup tworzyło swoje programy demonstracyjne składające się często tylko z samplowanej muzyki. Później coraz większy nacisk kładziono na grafikę i - w końcu - na poziom kodu oraz pomysły.

To właśnie dzięki dobrym pomysłom w ciągu ostatniego roku wiele grup powróciło do muzyki samplowanej, umiejętnie łącząc ją z dopracowaną grafiką i dobrymi pomysłami. Niektórzy sięgają także po zakurzone już dyskietki zawierające bardzo stare edytory do tworzenia tego rodzaju muzyki. W końcu również i mnie udało się znaleźć niezły, taki właśnie program, wraz z kilkoma przykładami utworzonych za jego pomocą kompozycji. Jest nim Sample mon v2.0, stworzony przez członków grupy DDG.

Aby poznać możliwości programu, proponuję wczytać jeden z plików, w nazwie którego znajdują się znaki $20 - są to dane o muzyce. Następnie należy załadować plik znajdujący się pod przed chwilą załadowanym przykładem. Są to dane dla Sample mon'a mówiące o kolejności odtwarzania odpowiednich kawałków. Teraz trzeba jeszcze władować główny program (Sample mon v2.0), uruchomić go i po użyciu klawisza P (Play) możemy przysłuchiwać się efektom wielogodzinnej cudzej pracy.

Program jest dość interesujący z tego względu, że maksymalnie wykorzystuje pamięć komputera zapisując każdy bajt w całości, a nie korzystając z tylko młodszych lub tylko ze starszych bitów. Dzięki tej operacji możliwe jest wykorzystanie dwa razy większej ilości pamięci.

Mam nadzieję, że przesłuchaliśmy już nagrane utworki i możemy przejść do głównego opisu.

Najpierw wyjaśnijmy sobie znaczenie kolejnych kolumn z liczbami:

plac	- adres, pod którym jest zapisana odpowienia linijka danych,
s1-e1	- początkowy oraz końcowy blok głosu pierwszego,
s2	- początkowy blok głosu drugiego,
*	- wybór ścieżki dźwiękowej:
1	- dolna ścieżka,
2	- dolna i górna ścieżka ,
3	- górna ścieżka,
0	- koniec utworu,
spd	- szybkość odtwarzania danej sekwencji, największa prędkość to $1E0 (oczywiście dane są wpisywane szesnastkowo),
con1,2	- wybór częstotliwości dla głosu pierwszego oraz drugiego, 0100 jest standardową częstotliwością (aby odkryć różne ciekawe efekty proponuję poeksperymentować wpisując w to miejsce różne wartości najlepiej z zakresu od $0001 do $0200).

Przez odpowiednie manipulowanie różnymi wartościami można osiągnąc bardzo wiele interesujących efektów (np. echo, wyciszanie dźwięku itp). Warto więc próbować samodzielnie, zapisując na kartce ciekawsze rozwiązania.

W trybie edycji utworu można również korzystać z kilku klawiszy służących do wyboru różnych opcji:

P	- odegranie aktualnie znajdującego się w pamięci utworu,
E	- wejście do trybu edycji (można użyć tylko Return),
X	- wyjście do interpretera basic'a,
+ ; -	- przewijanie danych na ekranie,
F1 ; F3	- zaznaczenie początku oraz końca obszaru danych,
Run/Stop	- zatrzymanie odtwarzania utworu,
Return	- wyjście z trybu edycji.

Będąc w edytorze, nacinięciem klawisza „strzałki do góry" przechodzimy do programu służącego do samplowania muzyki. Możemy z niego skorzystać jeśli posiadamy specjalny sampler podłączany do złącza user-port w C64. Program ten będzie w stu procentach działał z przystawką o nazwie Commodore Sfx Sound Sampler.

Oto opis opcji dostępnych z menu samplera:

F1	- wgrywanie do komputera pierwszej ścieżki (dolne cztery bity)
F2	- wgrywanie do komputera drugiej ścieżki (górne cztery bity)
F3	- odtwarzanie pierwszej ścieżki
F4	- odtwarzanie drugiej ścieżki
E	- zmiana początku oraz końca obszaru, do którego wgrywamy muzykę
M	- powrót do edytora utworu
@	- zamiana scieżek
L	- załadowanie danych z dysku
S	- nagranie danych na dysk

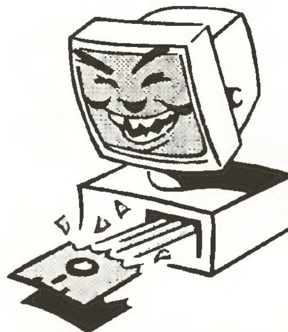
Jak widać, mimo swojej prostoty i stosunkowo niewielkiej długości, program Sample mon v2.0 ma całkiem pokaźne możliwości. Po niewielkich przeróbkach program powinien działać również z innymi (niż Commodore SfxSound Sampler) przystawkami, co zapewniło mu swego czasu sporą popularność.

Istotną zaletą pakietu jest osobny program, który pozwala na utworzenie procedury odtwarzającej gotowy utwór i wstawienie jej pod dowolny adres w pamięci C64. Cechą tej procedury jest stosunkowo czysty dźwięk mimo odtwarzania przy włączonym obrazie.

Życzę wszystkim miłej zabawy.

Jarosław "JARRI" Horodecki

P.S. Program ten znajduje się na 15 dysku PDP (o zasadach nabywania zestawów PDP piszemy na str. 16.).



SPRITE'Y i BASIC cz. 2

W poprzednim odcinku poznaliśmy podstawowe rejestry związane ze sprite'ami oraz napisaliśmy prosty program poruszający jednego sprite'a po ekranie. Dzisiaj poznamy kolejne rejestry odpowiedzialne za obsługę sprite'ów oraz spróbujemy stworzyć prostą grę opartą o tą technikę.

Na początek kilka słów przypomnienia. Jak wiemy C64 może wyświetlić najwyżej osiem sprite'ów. Za aktywność każdego z nich odpowiada rejestr znajdujący się pod adresem 53269. Pozycję poziomą i pionową kolejnych sprite'ów umieszczamy od adresu 53248 w kolejnych komórkach (najpierw X, a potem Y itd.). Za najstarszy bit pozycji X każdego ze sprite'ów (rozdzielczość pozioma wynosi 320 pikseli) odpowiada rejestr 53264. Wektory informujące o kształcie każdego sprite'a są usytuowane od adresu 2040 do 2047, a dane o jego kształcie zajmują 63 bajty.

Po tej krótkiej powtórce przystąpmy do poznawania dalszych rejestrów. Na pewno będziemy chcieli, aby obiekty w naszej grze były różnokolorowe. Aby dokonać zmiany koloru należy wpisać jego kod (tu można posłużyć się instrukcją dostarczaną wraz z komputerem) do rejestru koloru danego sprite'a. Rejestry te znajdują się w pamięci począwszy od adresu 53287 i jest ich osiem (czyli dla każdego sprite'a po jednym).Jak widać, każdy sprite może posiadać jeden niezależny kolor.

Czasem może się jednak zdażyć, iż dla uatrakcyjnienia szaty graficznej naszego programu zechcemy nadać każdemu obiektowi więcej niż jeden kolor. Otóż możemy tego dokonać przez włączenie trybu wielokolorowego wywietlania sprite'ów. Rejestr informujący o włączeniu tego trybu znajduje się pod adresem 53276 i ma identyczną strukturę jak rejestr włączający sprite'y. Czyli po wpisaniu do niego wartości 1 włączymy tryb wielokolorowy dla pierwszego sprite, natomiast wpisanie np. 6 spowoduje włączenie tego trybu dla drugiego oraz trzeciego sprite'a. Przy włączonym trybie wielokolorowym możemy wykorzystać w każdym ze sprite'ów oprócz jednego odrębnego koloru dodatkowe dwa definiowane globalnie dla wszystkich obiektów. Kody tych dwóch kolorów należy wpisać pod adresy 53285 oraz 53286. Pięknie!

Nie ma jednak róży bez kolców i przez włączenie trybu wielokolorowego rozdzielczość każdego sprite'a maleje do 12 na 21 pikseli przy zachowaniu tej samej wielkości. Tak więc każdy piksel staje się dwa razy dłuższy, co w niektórych przypadkach wygląda bardzo nieestetycznie. Jednak przez staranny dobór kolorów można temu zaradzić.

Aby móc rozpocząć tworzenie naszej obiecanej gry musimy jeszcze poznać rejestr znajdujący się w komórce 53278. Informuje on o kolizji sprite'ów przez zapalenie odpowiednich bitów. Gdy na przykład zderzą sie ze sobą sprite'y trzeci i ósmy to w komórce tej otrzymamy wartość 128+4=132. Ustawienie bitów jest analogiczne do rejestru trybu wielokolorowego oraz aktywności sprite'ów.

Poznaliśmy kilka nowych rejestrów, proponuję więc wykorzystać zdobyte wiadomości w praktyce.

Na początek napiszmy prosty programik:

```
10 FOR I=0 TO 63: POKE
   8192+I,255: NEXT
20 FOR I=0 TO 7
21 POKE 53287+I,1
22 POKE 2040+I,128
23 NEXT
30 POKE 53269,255
40 POKE 53264,0
50 FOR I=0 TO 15
51 READ X: POKE
   53248+I,X
52 NEXT
53 DATA 24,50,48,50,72,
   50, 96, 50
54 DATA120,50,144,50,168,
   50,192,50
```

Po jego uruchomieniu ujrzymy rządek spite'ów ustawiony w górnej części ekranu. Proponuję teraz poeksperymentować wpisując różne liczby w liniach 53 i 54, w których są zapisane kolejne pozycje poziome oraz pionowe sprite'ów. Zmieniając te pozycje będziemy również wpływać na stan rejestru kolizji, który możemy odczytywać instrukcją:

```
PRINT PEEK(53278)
```

Aby zaznajomić sie z trybem wielokolorowym możemy przełączyć wzory sprite'ów na adres 4096, przez zmianę wartości 128 na 64 w 22 linii i wprowadzić dodatkową linię włączającą tryb wielokolorowy:

```
60 POKE 53276,255: POKE 53285,5:
   POKE53286,9
```

Opanujcie dokładnie zmiany kolorów i przełączanie trybu wielokolorowego dla różnych sprite'ów.

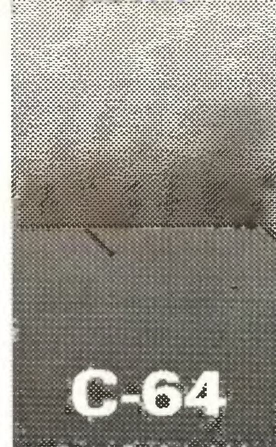
Gdy już dojdziecie do wniosku, że cały materiał teoretyczny opanowaliście do perfekcji to możecie zabrać się razem z nami do pisania pierwszej gry!

W naszej grze będziemy strzelać do poruszającego się obiektu. Nasz strzelec będzie się znajdował z lewej strony ekranu, a ruchoma tarcza z prawej.

Aby uprościć problem stworzenia ładnej szaty graficznej zakładamy, że strzelec jest schowany za ramką z za której wystaje jedynie strzała. Strzała i tarcza będą przedstawione za pomocą sprite'ów.

W naszej grze użyjemy tylko dwóch sprite'ów aby osiągnąć rozsądną szybkość działania programu. Zaczniemy oczywiście od wstawienia odpowiednich początkowych wartości do wszystkich potrzebnych nam rejestrów:

```
10 PRINT CHR$(147);: POKE53280,0: POKE53281,0
20 POKE 53269,3
30 POKE 53287,1: POKE53288,2
40 POKE 2040,128: POKE2041,129
50 POKE 53248,24: POKE53249,150
60 POKE 53250,255: POKE53251,150
70 FOR I=0 TO 127: READ X: POKE 8192+I,X: NEXT
80 DATA 0,0,0,0,0,0,0,0
81 DATA 0,0,8,0,0,12,0,128
82 DATA 14,0,192,15,0,224,15,240
83 DATA 255,255,254,255,255,255,255,255
84 DATA 254,224,15,240,192,15,0,128
85 DATA 14,0,0,12,0,0,8,0
86 DATA 0,0,0,0,0,0,0,0
```




```
87 DATA 0,0,0,0,0,0,0,189
90 DATA 3,16,0,3,184,32,3,252
91 DATA 96,3,254,224,3,255,160,3
92 DATA 255,0,3,254,0,3,254,0
93 DATA 3,254,0,3,254,0,3,254
94 DATA 0,3,254,0,3,254,0,3
95 DATA 254,0,3,254,0,3,254,0
96 DATA 3,254,160,3,254,224,3,252
97 DATA 96,0,184,32,3,16,0,189
```

W linii 10 kasujemy ekran i ustawiamy kolor ramki oraz ekranu na czarny. W lini 70 wczytujemy dane dla sprite'ów będących tarczą oraz strzałą (dane pobierane są z linii 80-99). Łatwo zauważyć, że używamy sprite'ów jednokolorowych, ponieważ nie kładziemy w tej chwili szczególnego nacisku na wygląd naszej gry, lecz na sposób jej zaprogramowania. Mam nadzieję, że wszyscy zdolni graficy będą w stanie - bez wiekszych problemów - dokonać w programie zmian, które umożliwiłyby wprowadzenie własnej grafiki. Przejdźmy jednak do zasadniczej części programu, czyli procedur zajmujących się ruchem obiektów i sprawdzaniem kolizji.

Nadamy teraz potrzebnym nam zmiennym odpowiednie wartości początkowe. Będą to licznik trafień oraz licznik wszystkich strzałów, dzięki którym będziemy w stanie ustalić wynik gry.

```
100 TR=0: OG=0: FL=0
```

Zmienne TR oraz FL to liczniki trafień oraz strzałów (zmienna FL przyda się nam nieco później). Warto jeszcze nadać odpowiednie wartości zmiennym odpowiadającym za pionowy ruch sprite'a-tarczy.

```
110 PY=150: AD=1
```

Zmienna AD będzie miała takie samo znaczenie jak zmienne A oraz B w naszym programiku z poprzedniego odcinka - będzie odpowiedzialna za kierunek ruchu sprite'a. Zmienna PY opisuje pozycję Y ruchomej tarczy.

Teraz rozpoczynamy tworzyć główną pętlę naszej pierwszej gry. Zaczniemy od wywietlania punktacji:

```
120 PRINT CHR$(19)"PKT:"TR"/"OG
```

Po uporaniu się z wyświetlaniem naszego wyniku, czas zająć się poruszaniem tarczy. Załatwi to za nas pętla :

```
130 PY=PY+AD: IF(PY=229)OR(PY=50)THEN AD=-AD
140 IF FL=0 THEN GET A$: IF A$=" "THEN PX=24:
    OG=OG+1: FL=1
150 IF FL=0 THEN 120
```

Przedstawione trzy linie są odpowiedzialne za ruch tarczy w pionie oraz za sprawdzanie, czy został naciśnięty klawisz SPACE, jeżeli tak to program wychodzi poza pętlę. Zwróćmy uwagę na zmienianą w linii 140 wartość zmiennej FL. Po wpisaniu do niej wartości 1 program będzie pomijał sprawdzanie klawiatury podczas ruchu strzały w kierunku tarczy. Jeżeli zmienna FL przyjmuje wartość 0 pętla programu kończy się już na linii 150.

Umiemy już poruszać tarczą. Można nawet uruchomić program, jednak po niaciśnięciu klawisza SPACE...

Brakuje oczywiście obsługi ruchu strzały.

```
160 POKE 53248,PX: A=PEEK(53278)
170 IF PEEK(53278)=3 THEN TR=TR+1: POKE 53248,24:
    FL=0: GOTO 120
180 PX=PX+2: IF PX255 THEN POKE 53248,24:FL=0:
    GOTO120
190 GOTO120
```

I oto już cała gra! Linia 160 to wpisanie nowej współrzędnej do rejestru pozycji X strzały oraz przyjęcie starej wartości rejestru kolizji sprite'ów. Jest to konieczne ze względu na dość specyficzne zachowanie tego rejestru (nie będziemy się teraz nim zajmować ze względu na konieczną znajomość systemu przerwań układu VIC oraz języka maszynowego).

Linia 170 sprawdza, czy pierwszy oraz drugi sprite zderzyły się. Jeżeli nie to dalsza jej część jest pomijana. Jeżeli tak to zwiększany jest licznik celnych trafień, a strzała wraca na swoje pierwotne miejsce z lewej strony ekranu. Podobnie dzieje się w linii 180, jednak licznik trafień nie jest zwiększany. Linia 190 to zakończenie pętli.

Natychmiast po uruchomieniu programu łatwo zauważymy jego podstawową wadę - szybkość jego wykonywania. Można oczywiście temu częściowo zaradzić przez próbę zoptymalizowania liczby instrukcji, czy też długości pętli, jednak nie rozwiązuje to w pełni problemu. Program nadal nie osiągnie zadawalającej prędkości działania.

Można także napisać cały program w języku maszynowym (ale tego jeszcze nie umiemy). Proponuję jednak najprostszą pętlę programu, sterującą pionowym ruchem tarczy, zastąpić bardzo prostą procedurą w języku maszynowym, wykonywaną na przerwaniach. Odciąży to program basic'owy od potrzeby ciągłego zmieniania pozycji Y tarczy oraz dbania o kierunek jej ruchu. Dzięki temu nasza strzała będzie mogła osiągnąć wiekszą prędkość, a gra stanie się nieco bardziej dynamiczna.

Aby wprowadzić do programu taką procedurę musimy dopisać kilka nowych linii zawierających kod:

```
1 FOR A=49152TO49190: READ X: POKE A,X: NEXT
2 DATA 162,11,160,192,142,20,3,140
3 DATA 21,3,96,238,3,208,173,3
4 DATA 208,201,228,240,7,201,50,240
5 DATA 3,76,49,234,173,11,192,73
6 DATA 32,141,11,192,76,49,234
```

Chcąc wykorzystać możliwości jakie daje ta procedura musiumy również napisać od nowa pętlę główną naszej mini-gry. Możemy zlikwidować zmienną FL, gdzyż pętla nie musi uwzględniać już ruchu tarczy.

```
100 SYS 49152: TR=0: OG=0
110 PRINT CHR$(19)"PKT:"TR"/"OG: GET A$: IF A$" "THEN
    110
120 PX=24: OG=OG+1
130 POKE53248,PX: A=PEEK(53278)
140 IF PEEK(53278)=3 THEN TR=TR+1: POKE53248,24:
    GOTO110
150 PX=PX+2: IF PX255 THEN POKE 53248,24: GOTO
    110
160 GOTO 130
```

Jak widzimy program nasz znacznie się uprościł. Zamiast specjalnych podwójnych pętli stosujemy tylko jedną instrukcję SYS 49152 w linii 100, która uruchamia naszą procedurę przerwania. Dalsza część programu nie wymaga już chyba wyjaśnień.

Po uruchomieniu programu spróbujmy zatrzymać go klawiszem RUN/STOP. Co widzimy? Otóż sprite nadal porusza się z prawej strony ekranu. Zatrzymać go możemy jedynie przez naciśnięcie kombinacji RUN/STOP+RESTORE. Warto również zwrócić uwagę na prędkość działania programu. Mylę, że jest to wszystko, co da się „wyciągnąć" z poczciwego języka basic. Można oczywicie „przerzucić" na język maszynowy inne fragmenty pętli, ale czyż wtedy nie prościej będzie napisać cały program w „maszynówce"?

Na razie proponuję rozbudowę naszej gry o dodatkowe możliwości, na przykład: ograniczony limit czasu, dopuszczalną ilość spudłowanych strzałów. Można wprowadzić także tablicę rekordów, ładną stronę tytułową i planszę gry. Jak widać możliwości jest bardzo wiele.

Mam nadzieję, że dzieki poznanym metodom obsługi sprite'ów napiszecie swoje własne, ciekawe gry.

Jarosław "Jarri" Horodecki

OPIS ZESTAWU PUBLIC DOMAIN PACK

nr 15 (dysk - marzec '92)

C-64

CONTACT DEALER v3.

Program ten z pewnością okaże się bardzo przydatny dla wszystkich prowadzących korespondecję z dużą ilością osób. Umożliwia wpisanie do pamięci komputera imion, nazwisk, ewentualnych pseudonimów, a także adresów i telefonów wszystkich znajomych. Całą bazę danych można oczywiście zapisać na dysku (program może współpracować nawet z dwoma stacjami).

Bez problemu możemy wydawać stacji dysków dowolne polecenia dzięki opcji Disk Command. Pozwala również dokonać wydruku naszej listy, sortować wszystkie adresy według kolejnych liter znajdujących się w danym rekordzie. Jeżeli wpisujemy adresy naszych znajomych ze sceny komputerowej, a składamy akurat demo to - w prosty sposób dzięki opcji Create greetings list - utworzymy listę pozdrowień gotową do natychmiastowego wstawienia do tekstu scroll'a. W celu odnalezienia konkretnego rekordu używamy opcji Search, która umożliwia poszukiwanie rekordu według zadanego klucza. Możliwa jest także bardzo prosta edycja wszystkich danych.

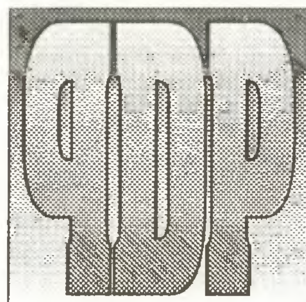
Beeftrucker v1.

Jest to prosty kompresor znakowy. Ze względu na zainstalowane systemy szybkiego ładowania i nagrywania oraz dość wygodną obsługę warto go dołączyć do swojej kolekcji.

Cross linker v3.

Kompresor znakowy nieco trudniejszy w obsłudze, jednak dający dużo większe niż Beeftrucker możliwości. Dzięki niemu można połączyć kilka oddzielnych kawałków danego programu, lub dane z kodem. Najczęściej tego typu programy używane są do łączenia czołówek z głównym kodem gry. Na uwagę zasługuje kilka ciekawych możliwości programu - n.p. można nim pakować zbiory znajdujące się od adresu $0200 aż do końca pamięci, czyli do adresu $FFFF, spakowany kod można umieścić pod niemalże dowolnym adresem, lub jako program uruchamiany z basic'a, można także włączyć tryb kodowania, w którym program wykonuje operacje EOR na całym kodzie przeznaczonym do spakowania. Podobnie jak Befftrucker, tak i CrossLinker jest wyposarzony w procedury szybkiego ładowania oraz nagrywania. Podczas pakowania można wybrać jeden z wielu efektów, które później będą towarzyszyć rozpakowywaniu.

PowerCruncer v7.1 grupy X-Rated.



Kompresor znacznie odbiegający od dwóch poprzednich (które były kompresorami znakowymi). Ten jest całkiem niezłym kompresorem sekwencyjno-bitowym. Efektywność jego pakowania jest porównywalna z najlepszymi kompresorami tego typu dostępnymi na C64. Program nie posiada niestety wbudowanych procedur szybkiego ładowania i nagrywania, jednak nie ma to większego znaczenia w sytuacji gdy średni czas pakowania wynosi około pół godziny.

Ciekawym rozwiązaniem jest wprowadzenie możliwości użycia napisu pojawiającego się pdczas dekompresji kodu.

Kolejny program użytkowy to edytor do tworzenia utworów muzycznych składających się tylko z muzyki samplowanej.

Sample mon v2.0.

Ze względu na pewną złożoność programu jego opisowi powięcamy osobny artykuł (na str. 7).

Handy Term v8.4.

Program obsługi modemu telefonicznego. Więcej informacji o nim możecie znaleźć w osobnym artykule powięconym „modemowaniu" na C64.

Jak zwykle, oprócz programów użytkowych zamieściliśmy także ostatnie numery polskich magazynów dyskowych dla C64. Jednym z nich jest znany już chyba wszystkim **Highlife**, którego wydawaniem zajmuje się obecnie nowa, polska grupa o nazwie Elysium. Mimo zmiany gospodarza magazyn nie zmienił niestety swojej już dość wysłużonej procedury obsługującej.

Drugim magazynem jest mniej znany tarnowski miesięcznik **Darkside**, wydawany przez sekcję tarnowską grupy Parados. Jest to młody magazyn, dlatego można jego autorom wybaczyć słaby poziom tekstów. Niedociągnięcia te w pełni jednak rekompensuje staranny program obsługi magazynu oraz dobre pomysły.

Warto także zwrócić uwagę na zrobiony „na wesoło" program Humor Basic. Jest to symulacja działania C-64 - komputer na wpisywane komendy odpowiada bardzo wesołymi komentarzami. Wprawdzie wszystkie napisy wyświetlane są w języku angielskim, jednak nic nie stoi na przeszkodzie, aby przetłumaczyć je na język polski (można tego łatwo dokonać, gdyż program napisany jest w języku basic).

Oprócz wymienionych umieściliśmy jeszcze kilka ciekawych programów demonstracyjnych, które z pewnością warto obejrzeć.

Jarosław "Jarri" Horodecki

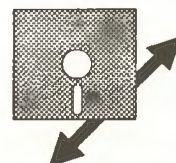
O zasadach nabywania naszych zestawów piszemy na str. 16. Poniżej przypominamy zawartość zestawów nr 13 i 14.

Zestaw nr 13

- Char Zoomer V3.1
- Colour Bar Editor V3.0
- Hires+A-FLI Editor
- FLI Designer V1.1
- Dirmaster V3.1
- Accesinus
- Music Routine Cruncher V1.5
- Gandalf Protector V3.0
- Gandalf Coder V1.0
- Disk - tape copy V2.0
- Gnd - packer V1.0

Zestaw nr 14

- LYNX XVI+
- SIDEBORDER LOGO EDITOR V1.0
- INTELPAINT
- FLIEDITOR V3.2
- 4*4 CHARMAKER
- F(R)ONT EDITOR 3
- THE GRAFIX-PACK II
- DEMA



„MODEMOWANIE"

C-64

Na łamach naszego pisma nie podejmowaliśmy jeszcze nigdy tematu telekomunikacji, czyli wykorzystania komputera do wymiany poczty elektronicznej, do „ściągania" najnowszych wersji różnych programów, czy też po prostu dobrej zabawy. Wszystko to możemy osiągnąć dzięki urządzeniu, które nazwano **modemem** (**mod**ulator /**dem**odulator). Urządzenie to służy do zamiany napływającego z komputera strumienia danych na sygnał, który może być przesłany przy użyciu zwykłej linii telefonicznej.

Przesył danych w tych liniach może być prowadzony z różnymi prędkościami. Najmniejsza - 300 bodów umożliwia otrzymanie przez komputer około 32.5 znaku na sekundę. Słowo około wynika z tego, że prędkość ta jest zwykle nieco mniejsza ze względu na zakłócenia, które niestety są stałym towarzyszem przesyłu danych polskimi łączami telefonicznymi. Nie radziłbym jednak nikomu pracować z tą prędkością, tym bardziej, że niektóre polskie BBS'y są dla niej zamknięte (nie wspominając już o zagranicznych...). Przeciętna prędkość, z którą można zwykle bez problemu pracować to 2400 bodów, co powinno dawać ok. 300 znaków na sekundę (w rzeczywistości rzadko kiedy osiąga się prędkość powyżej 250 znaków/sekundę).

Istnieją modemy pozwalające na pracę z prędkością 14400 bitów/sekundę (jednak realna prędkość pracy z tak szybkim przesyłem danych wynosi około 1500-2000 znakówna sekundę, powodem są zakłócenia w liniach telefonicznych).

Początkującemu użytkownikowi modemu polecam próby z prędkością 2400 bodów i - na początek - zabawę w granicach Polski, a jeżeli to możliwe to nawet tylko swojego miasta (dzwonienie do innych krajów może bardzo niekorzystnie wpłynąć na rachunek telefoniczny).

Warto zwrócić uwagę na dodatki do naszego modemu - mianowicie: sprzętowe pakowanie danych oraz sprzętową korekcję błędów. Te dwie opcje, a zwłaszcza ta druga, są bardzo przydatne w Polsce. Dzięki sprzętowej korekcji błędów komputer nie umieszcza na ekranie podczas pracy tak zwanych „krzaków", czyli znaków, które są odbierane w wyniku przekłamań powstałych na łaczach telefonicznych. Przy włączeniu sprzętowej korekcji błędów modem czeka, aż otrzyma prawidłową sekwencję znaków i dopiero wtedy wysyła ją do komputera. Powoduje to czasem zatrzymanie transmisji na krótką chwilę (jeśli zakłócenia są duże, modem musi czekać na dobrą sekwencję znaków).

Druga funkcja to sprzętowe pakowanie danych. Teoretycznie pakowanie danych o nazwie MNP5 umożliwia uzyskanie przesyłu danych z prędkością 9600 bodów (modemem 2400 bodów). Jest to jednak rzadki przypadek, gdyż prędkość ta może być osiągnięta tylko przy przesyłaniu nie spakowanych plików, czego - ze względu na koszt każdej minuty rozmowy - nie stosuje się. Tak więc w zasadzie pakowanie danych jest przydatne tylko podczas pracy z samym systemem, bez pobierania ani wywoływania danych.

Przy przesyłaniu dobrze spakowanych plików może się zdarzyć, iż sprzętowa kompresja spowoduje nawet spowolnienie transmisji.

Wspomniałem o czymś co nazywa się BBS. Co to właściwie jest? Otóż te trzy litery są skrótem angielskiego określenia Bulletin Board System.

BBS'y pozwalają wszytkim posiadaczom modemów na komunikację między sobą, wymianę korespondencji, czy też najnowszych wersji programów typu Public Domain. W Europie zachodniej istnieją również pirackie BBS'y, do których nie jest łatwo się dostać, a które oferują najnowsze gry i programy użytkowe (oczywiście nielegalnie). Na szczęście w Polsce prawdziwie pirackie BBS'y jeszcze nie istnieją, chociaż - z powodu braku praw autorskich - można również zauważyć w nich tytuły różnych gier i użytków nie będących programami Public Domain ani Shareware.

Co właściwie można zrobić, po połączeniu się z takim BBS'em. Po ukazaniu się na ekranie napisu CONNECT 2400 lub innego (w zależności od naszego modemu) zwykle ukazuje się strona tytułowa BBS'u. Z powodu braku możliwości wykorzystania grafiki wysokiej rozdzielczości (aby zachować zgodność z wieloma typami komputerów) strona ta jest zwykle złożona z szeregu znaczków semigraficznych (przy umiejętnym ich wykorzystaniu daje świetne efekty). Jeśli posiadamy odpowiedni terminal (program obsługujący modem) możemy włączyć grafikę ANSI, co uatrakcyjni naszą zabawę przez dodanie kolorów.

Następnie ukazuje się pytanie o naszą tożsamość. W legalnych BBS'ach odpowiadamy wpisując nasze prawdziwe imię oraz nazwisko. Jeśli danych nie ma na liście zarejestrowanych użytkowników, system automatycznie przydzielana nam status nowego użytkownika - po czym przekazuje szereg informacji na temat działalności danego BBS'u, jego możliwości oraz praw i obowiązków każdego użytkownika. Po przekazaniu często dość pokaźnej porcji informacji jesteśmy ponownie pytani o prawdziwe imię i nazwisko oraz o hasło jakiego będziemy używać. Hasło to jest potrzebne, aby inna osoba nie mogła zadzwonić i n.p. pobrać maksymalną ilość plików, wykorzystać wszystkie limity - podpisując się naszym imieniem, po czym wyjść z systemu nie ponosząc żadnych strat na swoim koncie. Dzięki wprowadzeniu haseł prawdopodobieństwo takiej sytuacji jest sprowadzone do minimum. Jedyną osobą, która jest w posiadaniu hasła oraz wszystkich innych danych, jest SysOp, czyli operator system (ang. SYStem OPerator).

Po podaniu naszego hasła musimy odpowiedzieć jeszcze na szereg pytań, które w każdym BBS'ie ustala sam sysop (mogą być one przeróżne: od pytań o adres, numer telefonu, czy wykonywany zawód, a skończywszy na ulubionych zwierzętach i ostatnio przeczytanej książce, czy też obejrzanym filmie).

Gdy odpowiemy już na wszystkie pytania ukazuje się jeszcze kilka informacji o systemie n.p.: lista osób, które wykonały najwięcej upload'ów oraz download'ów, lista dzwoniących do systemu w ciągu ostanich kilku godzin oraz wiele innych czasem bardziej, czasem mniej ciekawych informacji.

Każdy BBS jest wyposarzony w tzw. biuletyn, w którym sysop przekazuje wiadomości każdemu użytkownikowi. Uff! Wreszcie docieramy do menu nowego użytkownika.

Najczęściej sysop pozwala nowemu użytkownikowi tylko na zostawienie wiadomości i wyjście z systemu, a dobowy czas połączenia wynosi nie więcej niż 10-15 minut. Większy dostęp zyskujemy dopiero, po zatwierdzeniu nas przez szefa.

W sytuacji, gdy nasze dane budzą u sysopa wątpliwości może, w zależności od humoru, zostawić wiadomość będącą zarazem pierwszym ostrzeżeniem lub też (co jest częściej stosowane) po prostu skasować z BBS'u nowego użytkownika, aby miał on możliwość ponownego, prawidłowego, wpisania swoich danych, wypełnienia ankiety,

itp. Zakładamy jednak, że wpisaliśmy się prawidłowo i zostaliśmy zarejestrowani.

Każdy użytkownik posiada swoją rangę, która daje mu dostęp (lub też uniemożliwia) do różnych katalogów systemu, otrzymuje także określony limit czasowy, w którym musi zmieścić się z pracą w danym BBS'ie. Limit ten zwykle wynosi od około 45 do 90 minut. Każdemu użytkownikowi przydzielana jest również proporcja przysłanych plików do plików pobranych (tzw. UL/DL ratio). Proporcja ta określa ile będziemy mogli pobrać kilobajtów danych po **przysłaniu** jednego. Zwykle proporcja ta ulega zmianie w zależności od rangi użytkownika i jego działalności w danym BBS'ie. Warto też zapamiętać dość istoną zasadę: jeżeli już wyczerpaliśmy nasz limit, a koniecznie chcemy pobrać jakiś program, to pod żadnym pozorem nie wolno nam wysłać śmieci, po to tylko aby nabić sobie kilobajtów. Jak już wspominałem, każdy sysop ma władzę niemalże absolutną i jeżeli będziemy przysyłać stare bądź bezwartościowe programy sysop może nam nie zaliczyć upload'ów, dać ostrzeżenie lub nawet usunąć z listy użytkowników swojego systemu.

Gdy koniecznie chcemy coś przegrać, a nie mamy już limitu najlepiej jest wezwać sysopa (za pomocą odpowiedniej opcji) i powiedzić mu o naszej trudnej sytuacji. Sysop to też człowiek, więc powinien nam zwykle przydzielić te kilka kilobajtów. Jednak sysop może nie być akurat obecny, wtedy nic już nie możemy zdziałać...

Co jednak oprócz przesyłania plików oferuje nam BBS. Jest to oczywiście całkiem zależne od sysopa i jego pomysłów. Niektóre BBS'y oferują tylko niezbędne minimum możliwości, inne mają potężne, wielopoziomowe menu, z których możemy wybrać nieprzeliczoną ilość różnych wariantów.

Ostatnio bardzo modne stały się różnego rodzaju modemowe gry. Bardzo popularny był tzw. on-line tertris. Podczas grania w takie gry spokoju jednak nie daje poczucie, iż za każdą minutę naszej zabawy płacimy grube pieniądze. Oprócz gier sysopi często instalują różne inne atrakcje, n.p. różnego rodzaju konkursy (czasem nawet z nagrodami). Oczywiście każdy BBS ma dział poczty elektronicznej, w którym możemy zostawiać listy do różnych użytkowników, odbierać listy do nas, włączać się w ogólne dyskusje. Przykładem takiej dyskusji może być prowadzona ostatnio w warszawskich BBS'ach dyskusja na temat wyższości komputerów Commodore nad komputerami zgodnymi z IBM PC.

Jak widzimy działalność na BBS'ach daje wiele przyjemności bardzo uatrakcyjnia czas spędzony przy komputerze, a często, oprócz nowych wersji programów, możemy również korzystać z porad innych użytkowników, lub prowadzić z nimi interesujące dyskusje.

Coż, z pewnością wszytkich możliwości systemów BBS nie można opisać w tak krótkim artykule. To trzeba zobaczyć.

Jarosław "Jarri" Horodecki

OGŁOSZENIA

C-64

Twój komputer (ATARI, SPECTRUM, COMMODORE... itp.)stanie się źródłem dobrych dochodów (5-8mln zł). Te i inne informacje znajdziesz w poradniku (cena 25 tys. zł za zalicz. poczt.); oraz programy na C-64. Informacje (koperta+znaczek) Andrzej Prokopowicz, 12-100 Szczytno, ul. Chopina 15, woj. olsztyńskie.

Sprzedam Commodore C-64 II wraz ze stacją dysków 1541 II i osprzętem. Marek Kania, Al. Niepodległości 67/35, 02-626 Warszawa, tel. 48 89 58.

Literatura na AMIGĘ:
1. Opisy programów użytkowych, tom I - IV każdy po 39.000
2. opisy gier, tomy I - V - każdy po 39.000
3. Amiga Basic, tom I i II - 73.000
4. Amiga Amos, tom I i II - 69.000
5. Poznajemy komputer Amiga - 39.000
6. Amiga DOS - wprowadzenie do systemu - 39.000
Informacje - koperta+znaczek. Opłata przy odbiorze. Dolicza się koszt wysyłki. Ul. Kamieńskiego 76/2, 51-124 Wrocław.

Kupię PDP C-64 - dysk z kwietnia 1991r. Grzegorz Paździor, ul Aleji Rzeczypospolitej 6c, 59-220 Legnica.

SPRZEDAM: literaturę - komputery, elektronika i inne czasopisma: Bajtek (86-91), Top Secret, Radioelektronik itp. Spis - koperta+znaczek. M. Lis, Kochanowskiego 14B/20,01-864 Warszawa.

AMIGA 2000 - C-1MB ChipRam - 9.800.000, HardDisk QUANTUM 52MB, 17ms, 1,2 MB/s z kontrolerem do AMIGI 2000 - 7.000.000, Karta pamięci 8MB z 2MB FastRam do A2000 - 3.000.000. Genlock 2300 Pal - 2.000.000.
Krzysztof Lech, Niesiołowskiego 28/25, 87-100 Toruń, tel 48-22-58.

Sprzedam monitor RGB analog/TTL, 12", NEC. Sławomir Gasik, ul. M. Konopnickiej 1B/24, 99-100 Łęczyca.

LISTOWNY KLUB C-64, koperta+znaczek. 81-759 Sopot, ul. Bohaterów Monte Cassino 26m9.

Sprzedam tanio C+4, Datasette. R. Wołoszyk, ul. Karliczka 9/29, 40-488 Katowice.

LUDZIE! Klub Użytkowników C-64. Gruk Marcin, Rolnicza 74, 42-100 Zawiercie.

Sprzedam C-64, magnet., monitor, cartridge, joysticki, lit., programy i gry za ok. 2.500.00zl. Robert Roskosz, ul. Slowackiego 60/25, 80-257 Gdańsk, tel. 48-25-48.

AMIGA 500, COMMODORE 64 (TAŚMA-DYSK) WYMIANA - SPRZEDAŻ PROGRAMÓW. Koperta + znaczek. Paweł Witek, Karłowicza 45/55, 58-506 Jelenia Góra.

Amiga Club, INFORMACJA - koperta+znaczek. Ul. Wawelska 17/11, 44-217 Rybnik.

C-64 - zbiór map i opisów gier - sprzedam, info (znaczek): N. Wosik, 42-200 Częstochowa, ul. Słomskiej 11/39.

C-64, magnetofon, joystick, Final III, programy (300 sztuk), literatura - gwarancja 2 lata - 2.000.000. Tomasz Hatylak, Kochanowskiego 15/6, 38-500 Sanok, tel. 30-602.

Wymienię programy na Amigę. Tomasz Popławski, 15-836 Białystok, Ukośna 3/31.

Sprzedam nieużywaną drukarkę STAR LC10 COLOR do Commodore 64/128 za 1.500.000zł. Czarek, tel.: (02) 643701

KUPIĘ AMIGĘ 2000. Maciej Bryla, ul. Swobody 5, 44-230 Czerwionka, katowickie.

SPRZEDAM C-64 II komplet, literaturę, programy, tanio. M. Adamczyk, 42-633 Bytom, ul. Stcz. Bytomskich 242/8.

POLIGLOTA 2.2 - nauka angielskiego, niemieckiego; słówka i wyrażenia, wbudowany syntezator, cena 35.000zł. PANDA 1.4 - użytki matematyczne (wielomiany, funkcje, równania, układy) - 40.000zl; na przekazie podaj swoje nazwisko i typ klawiatury; otrzymasz programy i instrukcje. Tomasz Błaszczyk, ul Adamówek 1, 95-035 Ozorków (AMIGA).

SUPER DEMO (AMIGA) - 45.000. Pieniądze prześlij przekazem pocztowym. S. Golonka, ul. Wojska Polskiego 4B/4, 47-400 Racibórz

ASSEMBLER 6510 - lekcja 9

C-64

Mikroprocesor 6510 oprócz opisanych w dotychczasowych odcinkach naszego kursu rozkazów posiada jeszcze dużą ilość tak zwanych instrukcji niepublikowanych. Nimi to właśnie zajmiemy się w dzisiejszym odcinku.

Listy rozkazów niepublikowanych są dość trudno dostępne, gdyż mało kto naprawdę wie ile ich jest i co dokładnie każdy z nich robi. Mimo to postaram się przedstawić wszystkie znane mi instrukcje. Niestety nie jestem w stanie podać niektórych informacji o tych rozkazach (n.p. ilości cykli). Także wpisywanie tych rozkazów nie jest proste. Najlepszą metodą jest wpisywanie w programie odpowiedniej ilości instrukcji NOP, a następnie zastępowanie przy pomocy komendy M (w monitorze) odpowiednimi kodami. Można także posłużyć się specjalnym monitorem języka maszynowego, który posiada wbudowane kody tych instrukcji.

Rozkazy niepublikowane są na tyle specyficzne, że mogą nie działać na niektórych wersjach procesora 6510 lub nawet działać zupełnie inaczej niż jest to podane poniżej. Z tego powodu prosimy wszystkich czytelników, którzy zauważą pewne niezgodności o listy.

Po tym krótkim wstępie przejdźmy do opisu rozkazów. Większość z nich jest złożeniem prostych rozkazów, które były już opisane w poprzednich odcinkach.

mnemonik	kod
ALO #$nn	0B
ALO $nn	07
ALO $nn,X	17
ALO $nnnn	0F
ALO $nnnn,X	1F
ALO $nnnn,Y	0B
ALO ($nn,X)	03
ALO ($nn),Y	13

Rozkaz ten jest połączeniem dwóch rozkazów ASL oraz ORA. Najpierw procesor wykonuje przesunięcie zawartości akumulatora o jeden bit w lewo, a następnie, na otrzymanym wyniku operację logiczną ORA z podaym argumentem, który, po wykonaniu operacji, umieszczany jest w akumulatorze.

mnemonik	kod
RLA #$nn	2B
RLA $nn	27
RLA $nn,X	37
RLA $nnnn	2F
RLA $nnnn,X	3F
RLA $nnnn,Y	3B
RLA ($nn,X)	23
RLA ($nn),Y	33

Ten rozkaz stanowi połączenie ROL oraz AND. Podobnie jak ALO, tak i tutaj najpierw wykonywane jest przesunięcie zawartości akumulatora o jeden bit w lewo. Następnie na otrzymanym wyniku i podanym argumencie wykonywany jest rozkaz logiczny AND, wynik jest zapamiętywany w akumulatorze.

mnemonik	kod
LRE $nn	47
LRE $nn,X	57
LRE $nnnn	4F
LRE $nnnn,X	5F
LRE $nnnn,Y	5B
LRE ($nn,X)	43
LRE ($nn),Y	53

Jest to kolejne połączenie dwóch rozkazów. Tym razem są to rozkazy LSR oraz EOR. Najpierw wykonywane jest przesunięcie akumulatora o jeden bit w prawo, a następnie na otrzymanym wyniku jest wykonywana operacja logiczna EOR. Wynik zapisywany zostanie w akumulatorze.

menmonik	kod
RRA $nn	67
RRA $nn,X	77
RRA $nnnn	6F
RRA $nnnn,X	7F
RRA $nnnn,Y	6B
RRA ($nn,X)	63
RRA ($nn),Y	73

Rozkaz RRA stonowi połączenie rozkazów ROR oraz ADC. Najpierw wykonywana jest operacja przesunięcia akumulatora o jeden bit, a następnie do otrzymanego wyniku dodawany jest argument operacji. Wynik zapisywany jest do akumulatora.

mnemonik	kod
ANX $nn	87
ANX $nn,Y	97
ANX $nnnn	8F
ANX ($nn,X)	83

Rozkaz ten powoduje wykonanie operacji logicznej AND na akumulatorze oraz rejestrze indeksowym X. Wynik operacji jest wpisywany do podanej jako argument komórki pamięci.

mnemonik	kod
LAX $nn	A7
LAX $nn,Y	B7
LAX $nnnn	AF
LAX $nnnn,Y	BF
LAX ($nn,X)	A3
LAX ($nn),Y	B3

Rozkaz ten jest odpowiednikiem wykonania sekwencji instrukcji LDA oraz TAX. Najpierw argument ładowany jest do akumulatora, a następnie przerzucany do rejestru X.

mnemonik	kod
DCC $nn	C7
DCC $nn,X	D7
DCC $nnnn	CF
DCC $nnnn,X	DF
DCC $nnnn,Y	DB
DCC ($nn,X)	C3
DCC ($nn),Y	D3

Intrukcja ta jest połączeniem DEC oraz CMP. Najpierw wykonuje zmniejszenie argumentu o jeden, a następnie wynik tej operacji porównuje z aktualną zawartością rejestru akumulatora.

mnemonik	kod
ICS $nn	E7
ICS $nn,X	F7
ICS $nnnn	EF
ICS $nnnn,X	FF
ICS $nnnn,Y	FB
ICS ($nn,X)	E3
ICS ($nn),Y	F3

Rozkaz ten to połączenie rozkazów INC oraz SBC. Najpierw wykonywana jest operacja zmniejszenia zawartości komórki pamięci podanej jako argument o jeden, otrzymany wynik odejmowany jest od bieżącej zawartości akumulatora.

mnemonik	kod
ADL #$nn	4B

Rozakaz ten, podobnie jak dalsze opisywane w tym artykule posiada tylko jeden tryb adresowania - adresowanie natychmiastowe. Rozkaz ALD to połączenie rozkazów AND oraz LSR. Na początku procesor wykonuje pomiędzy akumulatorem a daną operację logiczną AND, a następnie otrzymany wynik przesuwa o jeden bit w prawo i umieszcza w akumulatorze.

mnemonik	kod
OAT #$nn	AB

C-64

Jest to najbardziej skomplikowany rozkaz procesora będącym połączeniem trzech innych rozkazów: ORA, AND oraz TAX. Na początku procesor wykonuje operację logiczną ORA z akumulatorem oraz liczbą $EE. Natępnie otrzymany wynik poddawany jest operacji AND z argumentem, a na końcu wynik umieszczany jest jeszcze w rejestrze X (pozostając także w akumulatorze).

mnemonik kod
SAS #$nn CB

Rozkaz ten na początku wykonuje operację logiczną AND na akumulatorze oraz rejestrze indeksowym X. Następnie od otrzymanego wyniku odejmuje podany argument. Na końcu otrzymany rezultat wpisywany jest do akumulatora.

mnemonik kod
TAA #$nn 8B

Wykonanie tego rozkazu powoduje przesłanie rejestru indeksowego X do akumulatora, a następnie wykonanie na tej wartości oraz podanym argumencie operacji logicznej AND. Wynik działania tego rozkazu zostaje zapisany w akumulatorze.

To już wszystko w dzisiejszym odcinku. Rozkazów niepublikowanych jest oczywiście nieco więcej jednak pozostałe są znacznie bardziej skomplikowane. Stanowią one zwykle połączenie kilku rozkazów i są raczej rzadko używane.

Warto jeszcze wspomnieć o tym, że najczęściej rozkazy te są używane tylko i wyłącznie po to, aby skutecznie zabezpieczyć program przed osobami niepowołanymi. Zwykle seria znaczków ??? pojawiających się przy deasemblacji kodu przy pomocy zwykłego monitora zniechęca nawet dobrego programistę.

Na koniec przedstawię propozycje rozwiązań zadań z poprzedniego miesiąca. Najpierw przypomnijmy sobie treść pierwszego z nich.

Należało napisać procedurę, która korzystając z arytmetyki dziesiętnej procesora sumowałaby wszytkie liczby od 0 do 99. Jak wynika z prostych obliczeń suma ta wynosi 4950, więc na zapisanie jej przy pomocy kodu BCD potrzebne są dwa bajty. Oto i moja propozycja:

```
LDA #$00     ;Wyczyszczenie komórek,
STA $FB      ;do których będzie zapisany
STA $FC      ;wynik działania.
SED          ;Włączenie BCD.
LDX #$00     ;Wyzerowanie rejestrów.
LDA #$00     ;
TAY          ;tutaj skok
BNE petla
CLC          ;Dodawanie kolejnych
ADC $FB      ;składników sumy do
STA $FB      ;aktualnego wyniku
LDA $FC      ;z uwzględnieniem
ADC #$00     ;starszego bajtu.
STA $FC      ;
TYA          ;Zwiększenie składniko
CLC          ;sumy o jeden.
ADC #$01     ;
INX          ;Zwiększenie indeksu o
CPX #$64     ;1 i sprawdzenie, czy już
BNE petla    ;koniec wykonywania pętli.
RTS          ;KONIEC
```

Zauważyliście pewnie, że nie można użyć rejestru indeksowego jako wartości składnika naszego dodawania. Dzieje się tak dlatego, że tryb arytmetyki dziesiętnej działa tylko dla operacji wykonywanych na akumulatorze. Rejestr X został użyty jako komórka kontrolna pętli.

Drugie zadanie polegało na wyświetleniu na ekranie kolejnych liczb od 0 do 9999, tak aby można je było odczytać. W moim rozwiązaniu opócz odpowiedniego opóźnienia wstawiłem sprawdzanie klawisza SPACE, którego naciśnięcie powoduje zatrzymanie odliczania. Oto program:

```
LDA #$00     ;Wyzerowanie licznika.
STA $FB      ;
STA $FC      ;
SED          ;Włączenie trybu BCD.
NOP          ;tutaj skok
BNE petla
LDA $FC      ;Ta sekwencja rozkazów
LSR          ;pobiera kolejne kawałki
LSR          ;czterobitowe licznika,
LSR          ;a następnie, by otrzymać
LSR          ;odpowiedni kod ekranowy
CLC          ;odpowiadający danej
ADC #$30     ;liczbie dodaje do
STA $0400    ;otrzymanej wartości
LDA $FC      ;liczbę $30 i tak
AND #$0F     ;otrzymaną wartość
CLC          ;zapisuje do czterech
ADC #$30     ;kolejnych komórek
STA $0401    ;pamięci ekranu znakowego,
LDA $FB      ;dzięki czemu w lewym
LSR          ;górnym rogu ekranu
LSR          ;widzimy kolejno
LSR          ;zmieniające się liczby
LSR          ;od 0 do 9999.
CLC          ;
ADC #$30     ;
STA $0402    ;
LDA $FB      ;
AND #$0F     ;
CLC          ;
ADC #$30     ;
STA $0403    ;
LDA $FB      ;Zwiększenie licznika
CLC          ;o jeden z uwzględnieniem
ADC #$01     ;możliwości przekroczenia
STA $FB      ;przez wynik liczby 99,
LDA $FC      ;gdy to nastąpi 1 dodawane
ADC #$00     ;jest także do starszego
STA $FC      ;bajtu licznika
LDY #$10     ;Petla opóźniająca.
LDX #$00     ;tutaj skok
BNE skok2
DEX          ;tutaj skok
BNE skok1
BNE skok1    ;
DEY          ;
BNE skok2    ;
LDA $DC01    ;tutaj skok
BNE skok3
CMP #$EF     ;Sprawdzenie klawisza
BNE skok3    ;SPACE.
LDA $FB      ;Sprawdzenie, czy licznik
CMP #$99     ;doszedł już do liczby
BNE petla    ;9999. Jeżeli nie to skok
LDA $FC      ;na początek pętli.
CMP #$99     ;Jeżeli tak to...
BNE petla    ;
CLD          ;Wyłączenie trybu BCD
RTS          ;oraz KONIEC.
```

Mam nadzieję, że komentarze w pełni wystarczą do zrozumienia działania programu. Proponuję dokonać małą zmianę w programie, która spowodowałaby odliczanie od 9999 do 0.

Ostatnie zadanie polegało na napisaniu procedur dodających i odejmujących liczby w kodzie BCD, ale wydaje mi się że przedstawianie rozwiązań tych problemów jest zbędne, ponieważ były one w porównaniu z dwoma poprzednimi zadaniami bardzo proste.

Jarosław "JARRI" Horodecki

PUBLIC DOMAIN PACK

PUBLIC DOMAIN PACK C - 64

Styczeń '91 (nr 1)

STRONA A

- Mega demo grupy „VISION"-MIST2

STRONA B

- Preview do gry „UN SQUADRON"
- Preview do gry „PUZZLENOID"
- Preview do gry „TURRICAN"

Luty '91 (nr 2)

STRONA A

- TUNE OF MONTH
- LOGO WRITER V 2.0
- FAST CRUELCRUNCH
- WRATH+(DEMO)[02]
- DREPTACZ _ BASIC

STRONA B

- SWISS CHEESE/CFA
- DISK FAST LOADER

Marzec '91 (nr 3)

STRONA A

- FONT GRUB 1.0
- PROJEKTANT DUSZKÓW
- STRZAŁKA 64+
- PIRATEK - GRA
- V4.0 - SYMPHONIES
- CRUISER
- THE FIRST
- COMMERCIAL BREAK
- RELAKATOR 64
- KOREKTOR 64
- FLASH

STRONA B

- HOT SHOT nr9 (zach. magazyn fanów)
- BAD NEWS nr2 - j.w.
- DEMO - rekord - 290 SPRITE'ów!
- DEMO: NEW INTRO
- DEMO: LET'S DYCP
- KONTAKT CORNER _ adresy, kontakty
- NEW FAST - działa z 1541 I 1541 II
- CSLINKER V2.0

Kwiecień '91 (nr 4)

STRONA A

- Digi - Organizer - program do tworzenia muzyki z użyciem digitalizacji dźwięku

STRONA B

- „ONE YEAR - RADIUS" - mega demo grupy RADIUS. Bardzo ładna grafika

Maj '91 (nr 5)

STRONA A

- CRUEL SOLIDERS - demo
- DESTINATION - demo
- SUCKER DJ! - demo (digi mix)
- MUSIC SEARCHER - do wycinania ilustracji muzycznych z programów

STRONA B

- MEGA DEMO „INFOSYSTEM 91"

Czerwiec '91 (nr 6)

STRONA A

- FONTEDITOR
- SINDATA EDITOR
- COLOR EDITOR
- DISK - NOTER
- GWIAZDY - demo graficzne
- FILGRAEPH 2.2/BML
- NOTE TO FLI V 2.2
- AFLI - EDITOR V 1.2
- RESET - MON,8,1
- TURBO - ASS 5
- ...HIGHLIFE #5
- AXEL NEWS #1
- DISK NOTKA/PADUA

STRONA B

- PSC - MAG #9'06/91
- CONSPIRE?OREGON - demo
- CONTACT DEMO/ORE
- SHOWPIX

Lipiec '91 (nr 7)

STRONA A

- Mega demo „MY, OH MY!" grupy LIGHT

STRONA B

- Game Music Composer - edytor muzyczny grupy GRAFFITY Węgier.

Sierpień '91 (nr 8)

STRONA A

- MegaDemo „Unnamed" grupy CAMELOT
- Sound Killer - edytor muzyczny grupy TOPAZ
- AFLI - Editor graficzny techniki A-FLI
- Disk-Dos obsługa komend stacji dysków
- Noter v2.2 grupy TOPAZ
- IFFL - Squeezer kompresor dyskowy
- Dismaster+ - edytor do dyskietek
- Super Copy - DOS szybki program kopiujący do zbiorów

STRONA B

- Mega Demo fińskiej grupy TOPAZ - „Graveyard Blues"

Wrzesień '91 (nr 9)

STRONA A:

- Mega Demo grupy FLASH

STRONA B:

- Hot Shot - magazyn dyskowy
- Code Sucker monitor - pr. użytkowy grupy PADUA
- Mountain Ride - gra w BASIC

Październik '91 (nr 10)

STRONA A I B:

- MEGA DEMO „AIRDANCE 4" grupy T.A.T.

Listopad '91 (nr 11)

STRONA A

- NEW LAW & ORDER!
- FLT/LEGOLAND
- FLT/LEGONOTE
- TERMINAT.2%/FLT

STRONA B

- SM. CRIMINAL #08
- SMALL BUT FINE
- HOLLY SMOKE/M12
- UNITE!/SYLVIO

Grudzień '91 (nr 12)

STRONA A

- ARMAGEDDON 3
- NOTE TO DEMO

STRONA B

- OUTRUN 2 MUS $ SFX
- AFTERBURNER/MON
- TRIVIA-GAME MUSIC
- FORM.I. SIMULATOR
- 2400AD END-TUNE
- NIGHTHUNTER DIGI
- ELIMINATOR MUSIC
- TOMCAT MUS./MON
- ZAMZARA TUNE/MON
- NOTE TO DISK
- HIGHLIFE#9

PUBLIC DOMAIN PACK AMIGA

Styczeń '91 (nr 1)

- Programy kompresorów danych
- Grafiki Borysa Vallejo
- Prezentacja najlepszych muzyczek
- INTUITRACKER

Luty '91 (nr 2)

- Request player; Multi ripper
- 3-rd day; Phantasmagoria - demo
- Master Seka; Virus Ekspert v1.6
- AMOS-programy; Moduły: Killing game show, Upon Me, Let's swing it.

Marzec '91 (nr 3)

- Najnowszy i najlepszy program muzyczny PROTRACKER V1.0 (pakiet programowy)
- Najlepsze muzyczki: NOW WAIT? - DR.AWESOME
- AMOS - procedury
- DEMO grupy REBELES „TOTAL TRIPLE TROUBLE"

Kwiecień '91 (nr 4)

- RUBBER VECTORS - demo
- KEFTALES - demo
- DISK MASTER V3.0
- Moduły muzyczne:
  > TECHNOSTYLE 2
  > GALAXY 2
- GRAFIKA - prezentujemy rysunki
  > RICK PARKS

Maj '91 (nr 5)

- VIRUS X 5.0
- VIRUS TERMINATOR
- PARADOX - demo
- STORMCHILD - demo
- Moduły muzyczne:
  > MIAMI VOICE
  > ANTI ATARI SONG

Czerwiec '91 (nr 6)

- POWER BOOT - własne menu dysku
- DISK CODING SYSTEM - program do zabezpieczania dysków
- Konwerter IFF - ANSI
- AUER NATION - demo
- Moduły muzyczne
- DOCS - opis gry ELWIRA
- LAMER DEFENCE - do wykrywania i niszczenia wirusów
- REWENG GO OF THE LAMER - grafika w trybie D_HAM

Lipiec '91 (nr 7)

- Sanity - demo
- Amiga - Tanx (1Mb) - gra
- Little Beau (1MB) - gra
- There is A Light/Tonid - modules

Sierpień '91 (nr 8)

- Real 3D - demo nowego programu do raytraceing'u
- Moduł Muzyczny XTC STEREO

Wrzesień '91 (nr 9)

- MODUŁY MUZYCZNE dla programu TFMX:
  > R - TYPE
  > THE HOUSE OF TECHNO
- VIRUS EXPERT v181 + 143
- BOOT BLOCK'I
  > BOOTX v 3.80
  > IMPLODER v 4.0

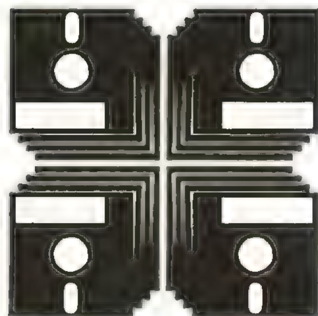
Październik '91 (nr 10)

- ANARCHY - „THE INSPIRATION IS NONE"
- DUAL CREW - „NEW DIMENSION"
- SANITY - „ELYSIUM"

PUBLIC DOMAIN PACK

Listopad '91 (nr 11)
- COMPUTER HEAD - animacja
- MODUŁY POD MEDPLAYER
  -CONFUSED
  -ROCKED
  I WIELE SAMPLI
- SAVE GAME „MONKEY ISLAND"

Grudzień '91 (nr 12)
- GEM X
- BOOTX V. 4.13
- FINAL KIT -monitor
- MEGA-MON
- VARIA



PUBLIC DOMAIN PACK C-64 TAPE NR 1

- TURBO
- SINUSDATA - EDITOR
- FAST CRUNCHER V3
- ANAL S.C. IBEYOND
- VECTOR - VICTORY
- PUZZLENOID+4
- TUNE OF MONTH #1
- NIM
- STRZAŁKA 64+
- LOGO - WRITER V.2.0
- CAN'T TOUCH IKUI
- NTRO PRV
- BONZIEED!!
- ZAX PACKIS
- READ THIS FIRST
- COMMERCIAL BREAK
- 290 SPRITES!
- NOTE - ABOUT
- BAD NEWS NR2
- TO BAD NEWS...
- CONTACT CORNER!
- PROJEKT DUSZKÓW
- SYMPHONY NR14
- SYMPHONY NR15
- SYMPHONY NR16
- SYMPHONY NR17
- SYMPHONY NR18
- SYMPHONY NR19
- CRUISER/GIANTS
- NOTE>ANO<PADUA
- LET'S DYSP!
- FINALTAPE
- MUSIC - SEARCHER

PUBLIC DOMAIN PACK C-64 TAPE NR 2

- TURBO
- PUBL. DOMAIN. INFO
- FONTGRUB 1.0
- DREPTACZ BASIC
- LOAD DIS FIRSY
- MACROASSEMBLER
- TURBOASSEMBLER
- RELOCATOR
- LOGOPAINTER 3!
- REASSEBLER
- SPRITE - EDITOR
- FAST - CRUEL U.2.5
- HIGHLIFE NR5
- AXEL NEWS NR1
- GWIAZDY
- FLIGRAPH 2.2/BML
- NOTE TO FLI V.2.2
- DISKNOTKA/PADUA
- MEGA PACKER/T
- MIST II/ VISION
- TTECHSCR &DYSP
- PLASMA - WORLD
- VECTORBOBS...
- VECTOR - PLOTS
- FLI - UPSCROLL
- BORDER - HIRES
- ROCK AROUND
- FACEWRITER
- CHAR EDIT 2+2
- DISKNOTER
- DESTINATION'91
- CONTACTDEMO/ORE
- FONTEDITOR
- THE END

PUBLIC DOMAIN PACK C-64 TAPE NR 3

- TURBO
- PUBLIC DOMAIN NOTE
- GRAVEYARD NOTES!
- NOTE FROM BEATI!
- ANONYM SPEAKINGI
- SNDK. V3.7/TOPAZ
- AFLI - EDITOR
- NOTER V2.2/TOPAZ
- DLW V1.5/TOPAZ
- CODE - S.MON/PADUA
- OPINION - POLL/PDA
- MOUNTAIN RAID
- PART 1
- PART 2
- PART 3
- PART 4
- PART 5
- FAIRLIGHT 1
- FAIRLIGHT 2
- FAIRLIGHT 3
- FAIRLIGHT 4
- FAIRLIGHT 5
- THE END

PUBLIC DOMAIN PACK C-64 TAPE NR 4

- TURBO
- OUT RUN 2 MUS & SFX
- AFTER BURNER/MON
- FORM.1.SIMULATOR
- 2400 AD.END - TUNE
- NIGHT HUNTER DIGI
- ELEMINATOR MUSIC
- TOMCAT MUSIX/MON
- ZAMZARA TUNE
- DYNAMIX TUNE
- HIGHLIFE NR9
- SNAKES C3
- SNARK C3
- SNERD C3
- WAREHOUSE C3
- STARTREK C3
- TOWER
- SNOOPY
- NEW LAW & ORDER
- FLT/LEGONOTE...
- TERMINAT.2%/FLT
- UNITE!/SYLVIO
- BALL - SCOPE/451
- TRIVIA - GAME MUS.
- RESET - MONITOR
- HOLY SMOKE

Zestawy „64 plus 4 PUBLIC DOMAIN PACK" można zamawiać wpłacając na konto: Bank PKO SA Oddział w Bydgoszczy konto nr: 5.09011-400522.7-2511-30-111.0 następujące kwoty: 20.000zł za pojedyńczy zestaw dyskowy dla C-64, 30.000 zł za zestaw programów PD na kasecie, 25.000zł za zestaw dla Amigi.

Blankiety wpłat powinny być CZYTELNIE wypełnione i zawierać: **imię i nazwisko, dokładny adres zamawiającego, skrót „PDP-64D"** - jeśli zamawiamy zestaw dla C-64 na dyskietce lub **„PDP-64T"** - dla zestawu taśmowego, zestaw dla Amigi prosimy zaznaczać skrótem **„PDP-A"** - dane te prosimy umieszczać na **wszystkich** odcinkach dowodu wpłaty.

W prenumeracie zestawy kosztują: **PDP-64 - 18.000zł** (12 numerów 216 tys. zł), **PDP-A - 22.000 zł** (12 numerów 264 tys. zł). Prenumeratę można zawrzeć w dowolnym terminie na okres od 3 do 12 miesięcy (do końca roku kalendarzowego). Powyższe warunki odnoszą się również do naszych zestawów wydanych w 1991r .

Zestawy taśmowe PDP-64 w 1992r. będą ukazywały się w miarę napływu nowych, ciekawych programów - o czym będziemy informować na łamach naszego pisma.

Zamów nie zwlekaj!

VOICETRACKER V4.0

C-64

Rewelacyjny program muzyczny!



Tylko 50.000 zł kosztuje fantastyczny edytor muzyczny wykorzystujący ogromne możliwości dźwiękowe komputera Commodore - 64. Oferowany zestaw zawiera dyskietkę lub taśmę magnetofonową z programem VOICETRACKER V4.0, instrukcję obsługi, oraz - dodatkowo - przykładowe demonstracje muzyczne. **UWAGA! Wersja magnetofonowa tylko 40.000 zł!**

Przedsiębiorstwo ABUK posiada wyłączność na dystrybucję tego programu. Wszelkie kopiowanie programu i powielanie instrukcji jest zabronione. Nabywcy otrzymują rejestrowane kopie programu wraz z prawem nabywania nowych wersji po znacznie obniżonych cenach oraz wymiany dyskietki w razie uszkodzenia. Studiom komputerowym proponujemy zakup hurtowy (przy zakupie powyżej 10 kompletów udzielamy 20% rabatu).
Chcąc stać się posiadaczem programu VOICETRACKER V4.0 wystarczy dokonać wpłaty 50.000 zł (wersja dyskowa) lub 40.000 zł (taśma) na konto: Bank PKO SA Bydgoszcz, konto nr: 5.09011-400522.7-136-11-111.0. Na blankiecie prosimy czytelnie podać swoje imię, nazwisko i adres wraz z dopiskiem „VV4.0" uzupełnionym literką „T" - taśma lub „D" - dyskietka.

W związku z pojawiającymi się kłopotami w dystrybucji oferowanych przez nas dyskietek i taśm (wynikającymi z nieczytelnego bądź niekompletnego wypełnienia blankietów wpłat) przedstawiamy obok specjalny druk. Blankiet ten może służyć jako zamówienie i dowód wpłaty dla wszystkich oferowanych przez nas usług: sprzedaż dyskietek i taśm PDP, Voicetracker'a, zamówienie ogłoszeń itd.

REDAKCJA



Odcinek dla wpłacającego

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Banku

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla posiadacza rachunku

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

Odcinek dla Poczty

Zł ....
słownie ....
....
wpłacający ....
....
....
(dokładny i CZYTELNY adres)

na rachunek:
**Przedsiębiorstwa ABUK sp. z o.o.
87-200 Wąbrzeźno, ul. 1 Maja 33,
Bank PKO SA Bydgoszcz, konto:
5.09011-400522.7-136-11-111.0.**

Opłata
zł..........

TREŚĆ ZAMÓWIENIA: ....

TREŚĆ ZAMÓWIENIA: ....

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA: ....

Prosimy o CZYTELNE wypełnienie.

TREŚĆ ZAMÓWIENIA: ....

Prosimy o CZYTELNE wypełnienie.

Zapraszamy wszystkich do udziału w stałym konkursie pod hasłem:

Najlepszy program miesiąca

W konkursie udział mogą brać wszyscy, którzy nadeślą własne, nigdzie nie publikowane prace. Tematyka programów dowolna. Konkurs rozgrywany jest osobno dla komputerów C-64 i Amiga. Teksty programów należy nadsyłać na adres redakcji na dyskietce lub w postaci czytelnego rękopisu (dyskietki będą przez redakcję zwracane). Objętość programu wraz z opisem i komentarzem nie powinna przekraczać 4 stron maszynopisu. Raz w miesiącu Sąd Konkursowy wybierze najlepsze programy przyznając ich autorom dwie główne nagrody po

500.000 zł każda.

Decyzje Sądu Konkursowego są nieodwołalne. Oprócz zdobycia głównej nagrody autorzy mają szansę na publikację swych prac na łamach naszego pisma. Pracę prosimy podpisywać imieniem i nazwiskiem oraz dokładnym adresem autora.

Redakcja

WSZYSTKICH ZAINTERESOWANYCH NABYCIEM ZALEGŁYCH NUMERÓW

„64 plus 4 & AMIGA"

INFORMUJEMY, ŻE POSIADAMY JESZCZE OGRANICZONĄ ILOŚĆ NUMERÓW OD LISTOPADA 199OR. DO GRUDNIA 1991R.

ZAMÓWIENIA PROSIMY KIEROWAĆ NA ADRES :
Przedsiębiorstwo ABUK sp. z o.o.,
87-200 Wąbrzeźno,
ul. 1 Maja 33.

(Pod tym adresem mieści się dział kolportażu - tam też prosimy przesyłać wszelką korespondencję dotycząca kolportażu czasopisma, dyskietek, taśm itd. Adres redakcji się nie zmienił - patrz stopka.)

PARTY W STOLICY

DLA WSZYSTKICH

Po nie tak dawno mających miejsce dwóch imprezach w Trójmieście w końcu także i mieszkańcy stolicy zdecydowali się zorganizować spotkanie commodorowców. Przygotowania trwały dość długo, po całej Polsce krążyły zaproszenia wraz z małą mapką wskazującą dokładne miejsce, w którym miała się odbyć impreza.

Organizatorzy tym razem połączyli spotkanie użytkowników AMIGI i jej starszego brata - C64.

Data spotkania została wyznaczona na 28 oraz 29 marca. Party to, jako pierwsze w Polsce miało się odbywać przez całe dwa dni non-stop.

Organizatorzy zapewnili wiele atrakcji: bufet z różnymi napojami oraz ciastkami, konkursy. Wszystkie dema, obrazki były wyświetlane na dużym ekranie ustawionym przed schodami (z powodzeniem używane jako widownia). Muzyka była słyszalna w całej szkole dzięki dobrej jakości nagłośnienia.

Oczywicie, jak przystało na prawdziwy konkurs były też atrakcyjne nagrody: za najlepsze demo - ostatnia wersja modułu ActionReplay, za najlepszą muzykę - wysokiej jakości sampler stereofoniczny i za najlepszy obrazek - program Deluxe Paint IV. Wszystko dla amigowców. Niestety użytkownicy C64 nie mogli zdobyć żadnych nagród gdyż sponsor nie wywiązał się ze swoich obietnic.

Na party zjawiło się bardzo dużo ludzi z całej Polski. Z bardziej znanych zjawiły się grupy (amigowskie): Alchemy, Action Direct, Deform, Katharsis, Joker, W.F.M.H., Old Bulls i kilka innych. Większość z nich zaprezentowała swoje prace we wszystkich konkursach. Przy czym największym powodzeniem cieszył się konkurs na najlepszą muzykę. Przesłuchanie trwało dobrych kilka godzin. Niestety poziom prezentowanych utworów był dość niski, tak więc mało kto był w stanie wytrzymać do końca . Najlepszy okazał się utwór RAF'a z dobrze znanej w Polsce grupy Katharsis.

Dużo krócej trwał konkurs na najlepszą grafikę. Prace były przeglądane dość szybko, a ich poziom, za wyjątkiem kilku, był bardzo dobry. W tym konkursie zwyciężył RYS z grupy Joker.

Kulminacyjnym punktem programu był konkurs na najlepsze demo. I tutaj wystąpiły pewne problemy natury technicznej. Mianowicie oprócz jednego dema, żadne nie chciało się uruchomić na Amidze 2000. Podczas zmiany komputera wszyscy mieli okazję obejrzeć demo graficzne grupy SILENTS nagrane na taśmie magnetowidowej, a stworzone przy użyciu profesjonalnego sprzętu audio-video. Po krótkiej przerwie mogliśmy obejrzeć pozostałe dema.

Najlepszym z nich okazała się być praca grupy Deform, drugie miejsce zajęła grupa Alchemy, a trzecie W.F.M.H.

Wyniki wszystkich konkursów były ustalane na podstawie specjalnie przygotowanych kart do głosowania, na których należało wpisać numery najlepszych, naszym zdaniem, utworów, obrazków i programów demonstracyjnych. Aby nie było mowy o stronniczości, głosujący nie znali autorów prezentowanych prac. Niestety z demami nie można postąpić w podobny sposób i wyniki konkursu w tej kategorii nie były chyba zbyt obiektywne.

Teraz kilka słów o C64.

W odróżnieniu od spotkania w Gdyni, do Warszawy przyjechało sporo osób. Obecne były następujące grupy: Charged, Taboo, Elysium, Parados, Skylight, Axel oraz Cavem. Na C64 w zasadzie nie odbył się żaden konkurs. Pierwszym powodem był z pewnością brak nagród, a drugim całkowity brak zainteresowania tą maszyną za strony amigowców (co mnie nie dziwi) oraz brak zainteresowania samym konkursem ze strony commodorowców (co mnie dziwi). Tak więc po podłączeniu C64 do dużego ekranu zebrała się przed nim bardzo mała grupka, która obejrzała kilka „demek" i obrazków.

Wszystko to jednak w porównaniu z Amigą prezentowało się dosyć słabo z powodu dużo gorszej grafiki, muzyki i pomysłów.

Po kilkunastu minutach C64 został wyłączony i rozpoczęła się projekcja filmu "Rzecz", podczas którego niektórzy rozpoczęli masowe kopiowanie dysków, inni zdecydowali się na odpoczynek (niekiedy w dość dziwnych pozach - ręce na klawiaturze, monitor włączony itp.).

Około godziny drugiej nad ranem ruch na party jakby ustał, większość uczestników zaczęła już odczuwać coraz wyraźniej brak snu. Jednak na drugi dzień rano wszyscy ruszyli do pracy ze zdwojoną energią. Niektórzy próbowali nawet rozpocząć zawody w rzucie dyskiem, jednak organizatorzy uspokoili zawodników.

Zasadniczo drugi dzień spotkania przeszedł na spokojnych rozmowach i kopiowaniu.

Podsymowując należy stwierdzić, iż orgranizacja imprezy była dobra, uczestnicy również dopisali, choć dobrze by było, gdyby zjawiła się wieksza grupa miłośników C64. Póki co widać na polskiej scenie bardzo wyraźną przewagę amigowców nad "małymi" commodorowcami. Poza tym scena amigowska jest jakby ciekawsza i doroślejsza.

Jarosław "Jarri" Horodecki

AMIGA

UTOPIA

Nareszcie jest! Od dawna oczekiwana i wytęskniona Utopia. Wszyscy fani gier w rodzaju SimCity, Megalomania czy Moonbase do boju! Nowa propozycja Gremlina jest naprawdę interesująca. Muszę tu od razu zaznaczyć - wszyscy „strzelacze", nie czytajcie dalej tego artykułu. Ta gra wymaga MYŚLENIA, PLANOWANIA i PRZEWIDYWANIA!

Pomysł jest zaczerpnięty z nieśmertelnego SimCity, jednak w Utopii znajdujemy o wiele więcej ciekawych możliwości. Dowodzimy młodą, nieokrzepłą kolonią z Ziemi. Jest rok 2090, zostawiają nas na powierzchni jakiejś planety z kupką budynków i garścią pieniędzy. Jest to bardzo niewiele. (Uwaga, posiadacze pudełka Action Replay: po uruchomieniu gry liczba pieniędzy „jest" w komórce $135A6 (długie słowo), wystarczy zmienić jej wartość i już problem gotówki mamy z głowy). Nasz żywot nie jest już tak sielankowy jak w SimCity, musimy walczyć z kolonią innej rasy. Często zdarza sie, że twoich ludzi atakuje wirus (nie komputerowy), brakuje prądu, tlenu, pożywienia. Od czasu do czasu jakiś nawiedzony terrorysta wysadzi ci w powietrze budynek, albo dwa; za chwilę spadnie ogromny meteoryt, niszcząc połowę kolonii, a potem obcy zrobią z twojego osiedla płaską płaszczyznę (gryps nauczycielski!). Tak może wyglądać Twoja gra w Utopię. Może,ale nie musi.

My zrobimy wszystko aby wydarzenia potoczyły się całkiem inaczej. Po uruchomieniu czeka nas krótkie (no, powiedzmy) intro, po którym wybieramy opcje gry. Pojawia się 9 ikon:

1.1 Tutor, czyli trening: zero wrogów i fura pieniędzy.
1.2 LOAD (odczytanie stanu gry z dysku).
1.3 SAVE (zapisanie stanu gry na dysk).
2.1 Wczytanie muzyki. (Te delicje (o boże!) tylko dla szczęśliwych aMEGAntów).
2.2 Wybranie scenariusza.
2.3 Formatowanie savegame-dysku.
3.1 FX'y on/off.
3.2 Muzyka on/off.
3.3 Exit (do dzieła!).

Największym moim problemem w czasie gry w Utopię, było przeznaczenie budowli. Na początku nie wiedziałem co do czego, ale teraz już wiem:

LIVING QUARTERS 2000 Gr. - domy mieszkalne dla ludzi; należy uważać aby gęstość zaludnienia ne przekroczyła 100, ponieważ zwiększa to śmiertelnośc i niezadowolenie mieszkańców.

FLUX POD 3000 Gr. - przekaźnik energii. Wszelkie inne budynki można stawiać tylko w jego pobliżu.

HYDROPONICS 2500 Gr. - służą do produkcji żywności, wymagają dziesięciu ludzi obsługi.

STORE 2000 GR. - magazyny. Należy uważać, aby zawsze mieć trochę wolnego miejsca w magazynie, gdyż jego brak spowoduje zatrzymanie produkcji uzbrojenia (brak miejsca na rudę z kopalni).

CHEMICAL PLANT 8200 Gr. - służy do produkcji paliwa niezbędnego dla statków kosmicznych (oprócz Fusion cruiser'a).

RADAR 2500 Gr. - radar.

MINE 5000 Gr. - kopalnia, służy do wydobywania rudy (Ore), bez której produkcja czołgów i kosmolotów jest niemożliwa.

LASER TURRET 5000 Gr. - działo laserowe, skuteczne przeciwko wszystkim siłom nieprzyjaciela.

MISSILE LAUNCHER 3000 Gr. - wyrzutnia rakiet. Uwaga! Wymaga ręcznego odpalenia!

SOLAR PANEL 800 Gr. - bateria słoneczna, niestety czasami nie działa z powodu częstych zaćmień słońca.

HOSPITAL 8500 Gr. - szpital; zmniejsza podatność populacji na wirusy, a także pozwala na kontrolę liczby urodzin (birth rate). Birth rate: NONE - zero przyrostu naturalnego, LOW - niska liczba urodzeń, MEDIUM - średni poziom, HIGH - wysoki przyrost ludności.

ARMAMENT LABORATORY 6000 Gr. - służy do produkcji broni, potrzebnej podczas produkcji uzbrojenia.

FUEL TANK 3000 Gr. - zbiornik na paliwo dla statków kosmicznych.

LABORATORY 5000 Gr. - laboratorium; jeżeli posiadamy dużo laboratoriów i odpowiednio je dotujemy to po pewnym czasie doczekamy się, „epokowych" wynalazków i podwyższenia naszego poziomu technicznego (wpływa to na jakość uzbrojenia).

SECURITY 4000 Gr. - kolonialna policja, czasem niezbędna.

TANK CONSTRUCTION YARD 6000 Gr. - fabryka czołgów, potrzebuje rudy i broni, przy maksymalnej (10 osób) obsadzie produkuje jeden czołg miesięcznie.

WORKSHOP 2800 Gr. - warsztat; produkuje rzeczy potrzebne ludziom (drobiazgi); możemy nimi handlować.

SHIP CONSTRUCTION YARD - fabryka statków kosmicznych, potrzebuje tego samego co TANK YARD, plus lądowisko w pobliżu. Produkcja nie odbywa się jednak automatycznie. Po zbudowaniu obiektu należy wybrać typ statku do konstrukcji. Czas budowy jest zależny do siły statku i jego wielkości. Statki:

EXPLOLER - buduje lądowiska z powietrza, NIGDY nie ginie na terytorium przeciwnika.

FIGHTER - myśliwiec, słaby, ale szybko się buduje i nie wymaga dużo paliwa.

ASSAULT CRAFT - statek ten (wygląda jak bombowiec) służy do przełamania wrogich lini obronnych.

CRUISIER - krążownik, o wiele silniejszy od ASSAULT CRAFT'a.

WARSHIP - pancernik „mocarz przestworzy" nie do zestrzelenia (teoretycznie).

LAUNCHPAD 3000 Gr. - lądowisko dla statków, możemy zlecic jego budowę EXPLORER'owi.

COMMAND CENTRE 9000 Gr. - Stanowisko dowodzenia, bez niego kompletnie przestajesz się orientować w sytuacji.

LIFE SUPPORT 7000 Gr. - fabryka tlenu.

POWER STATION 10000 Gr. - elektrownia; o wiele bardziej wydajna i niezawodna niż SOLAR PANEL'e.

SPORTS COMPLEX 6000 Gr. - stadion, umożliwia przeprowadzenie igrzysk, bardzo podnoszących morale kolonistów. (Uwaga: podczas trwania igrzysk, nie można nic budować, gdyż uczestniczą w nich wszyscy mężczyźni). O ile zapędzisz jajogłowych do roboty, to po pewnym czasie doczekasz się np. SPACE MOSS CONVERTER (to samo co LIFE SUPPORT, ale tańsze i mniejsze).

LAND MINE - mina przeciwczołgowa (twoje czołgi są bezpieczne).

LONG DISTANCE RADAR - radar o zwiększonym zasięgu.

FUSION CRUISER - krążownik o napędzie atomowym.

I wiele,wiele innych. Wszystkie te wynalazki, bardzo pomagają w śmiertelnych zmaganiach z Obcymi, należy więc nie szczędzić środków na badania. Gra bardzo wciąga, aby wygrać należy spędzić nad Utopią parę ładnych godzin, ale... na pewno nie będziesz się nudził, czego życzy Ci

AMBER.

Mega-Lo-Mania

AMIGA

Gdzieś w przestrzeniach wszechświata jest gigantyczna, przeźroczysta kula. Tam rodzą się planety i ciała niebieskie wypychane w pustki kosmosu. Dla niektórych niefortunnych planet, ewolucja kończy się wytworzeniem inteligentnej formy życia. Wtedy zaczynają się zawody pomiędzy półbogami i wyzwalane są wszystkie moce wszechświata. Każdy z nich chce zdobyć absolutną kontrolę. Powoli ogromna kula zostaje otoczona przez wszystkie rodzaje istot. Większość z nich musiała przebyć wiele lat świetlnych, aby zobaczyć ten boski pojedynek, pojedynek będący częścią formowania boskiego świata. Każdy z półbogów chce zostać panem lub bogiem i rządzić biednymi i nieświadomymi ludźmi. To gra. Gra ewolucji, destrukcji i siły. To gra w Mega Lo Manię...

No cóż, tym razem Ziemia nie miała szczęścia i 9500 lat przed naszą erą zaczęła się straszliwa i zawzięta walka półbogów.

Na początek - wybór boga. Mamy cztery możliwości.

1. SCARLET - porywcza i agresywna.Półbogini Plejadów. Kontroluje czerwonych ludzi.

2. OBERON - zdradliwy i bezlitosny,na wpół uznany król Algoli. Kontroluje żółtych ludzi.

3. CAESAR - mściwy i nieobliczalny, ojciec chrzestny mafii Trapezian. Kontroluje zielonych ludzi.

4. MADCAP - podstępny i śmiertelnie niebezpieczny. Największy najemnik andromedański.

Wyboru dokonujemy klikając na podobiznę wybranego boga. Uff...

Teraz należałoby przyjrzeć się opcjom programu (OPTIONS).

1. AUTO SLOW ON/OFF - opcja włączona - powoduje automatyczne przełączanie upływu czasu na najwolniejszy, gdy atakowany jest nasz własny zamek.

2. SPEECH ON/OFF - włącza lub wyłącza znakomite digitalizacje ludzkiej (?) mowy.

3. MUSIC ON/OFF - włącza lub wyłącza muzykę (tylko podczas gry).

4. SFX ON/OFF - efekty specjalne (dźwięk). Opcja włączona powoduje konieczność żonglowania dyskietkami, więc jeśli nie masz dodatkowej stacji dysków to radzę jej nie używać.

5. NEW GAME - zaczynamy od poczatku, ale nie można ponownie wybrać władcy.

6. LOAD/SAVE - wbrew pozorom, opcje te nie mają nic wspólnego z dyskiem. SAVE podaje kod do aktualnego etapu (uwzględniając władcę oraz ilość ludzi), LOAD pozwala na wpisanie kodu.

Let's start! Teraz należy wybrać jedną z trzech wysp, kliknąć na PLAY ISLAND, przydzielić odpowiednią ilość ludzi i wybrać miejsce na zamek, klikając na którymś z sektorów małej mapki.

Zaczynamy grę.

Ekran podzielony jest na trzy części. Największa przedstawia obraz wybranego sektora. Po lewej stronie na górze znajduje się mała mapka , dzięki której orientujemy się w sytuacji. Klikając na dowolny sektor zmieniamy zawartość głównego okna. Aktualny sektor jest oznaczony na małej mapie ramką. Małe tarcze obok mapki oznaczają władców. Jeżeli oprócz ciebie walczy jeszcze dwóch lub trzech innych bogów, to możesz zawierać sojusze, klikając lewym klawiszem na jedną z tarcz. Cyferki pojawiające się obok tarczy oznaczają ilość ludzi danego władcy w aktualnym sektorze. Klikając na dowolną z cyferek lewym przyciskiem, zamiast mapki pojawia się dokładny skład armii. Nad tarczami jest ikona przedstawiająca ludzika. Służy do sterowania upływem czasu. W lewej dolnej części ekranu znajduje się menu. Na początku jest ono niewielkie, ale wkrótce ilość opcji się zwiększy.

Ikonomania.

Górny rząd od lewej: wykaz zaprojektowanych rzeczy i zużycie materiałów; informacja o stanie zniszczenia zamku i innych budowli. Jeżeli masz zaprojektowaną tarczę, to klikając na nią w tym menu, następnie na którąś z budowli, naprawiasz ją. Następna ikona pozwala nam sprawdzić ilość lub dostępność broni defensywnych, a także pozwala zainstalować daną broń. Ostatnia ikona w górnym rzędzie służy do tworzenia armii i jej wyposażenia. Gracz może posiadać tyle armii, ile jest sektorów na mapie, ponieważ dwie jego armie na tym samym sektorze zostaną automatycznie połączone. W armii może sie maksymalnie znajdować do 250 żołnierzy.

Drugi rząd od lewej: ikona pierwsza służy do projektowania broni. Im więcej ludzi jest zaangażowanych do projektowania, tym szybciej dany projekt zostanie opracowany. Najlepsze bronie znajdują się na dole i ich opracowanie trwa najdłużej. Czas projektowania broni może się zmniejszyć, gdy uzyskamy skok ewolucyjny.

Reszta ikon jest bardzo zróżnicowana, zależnia od etapu, miejsca usytuowania zamku, itp. Są to ikony służące do przydzielania ludzi do wydobywania surowców. Pojawią się też po kolei trzy ikony służące do budowy kopalni, fabryki i laboratorium. Do każdej z nich trzeba przydzielić odpowiednią ilość ludzi. Im jest ich więcej, tym szybciej zbudują.

Dzięki fabryce możemy wyprodukować bardziej skomplikowane przedmioty (pojawi się dodatkowa ikona). Laboratorium służy do projektowania skomplikowanych broni: samoloty, rakiety, statki kosmiczne, lasery itp. Aby wygrać, należy jak najszybciej uzbroić (dobrze!) jak największą armię i znieść przeciwników z powierzchni ziemi (łatwo powiedzieć!).

Na zakończenie kilka rad, oraz kodów:

1.Ludzi można oszczędzać do następnej epoki, czasem jest to niezbędne do zwycięstwa.

2. Staraj się pozwolić twoim ludziom najpierw rozmnożyć się, a dopiero potem mając ich ok. 30-40 zatrudniaj do różnych prac.

3. Zaraz po wyprodukowaniu dobrej broni atakuj z największą możliwą siłą.

4. Staraj się jak najszybciej zawiązać jakiś korzystny sojusz.

5. Obserwuj rozwój ewolucyjny innych bogów.

Władca - Oberon
KGPCIDQGIHG - 2 epoka, 170 ludzi.
JKYBAHHWSHQ - 3 epoka, 225 ludzi.
EWZCUTVCCOO - 4 epoka, 265 ludzi.
KDXAERFUFCA - 5 epoka, 300 ludzi.
IIXAWSLUFCW - 6 epoka, 300 ludzi.
QGKAUDYKLUG - 7 epoka, 305 ludzi.
KGDAWWLPMBG - 8 epoka, 200 ludzi.
JHNAULUSFTC - 9 epoka, 140 ludzi.
Firma: I mageworks
Team: Sensible Software
Programowanie: Chris Chapman
Grafika: Jon Hare
Muzyka: Richard Joseph

Opis: Michał Gosztyła (Amber)

AMIGA

PAKUJEMY CHŁOPAKI
czyli „rozpiska" PowerPacker.library

PowerPacker stał się pewnym standardem i jest to zasługa prostoty obsługi tego programu w połączeniu z jego efektywnością oraz szybkością. Prawie każdy nowy program jest przystosowany do odczytu danych spakowanych przy użyciu PowerPacker'a. Jeżeli jednak chcemy użyć spakowanych danych w naszym programie to chcemy czy nie chcemy musimy je rozpakować. Istnieją tu dwie metody: pierwsza - prostsza - to użycie programu źródłowego do rozpakowywania danych, druga to użycie biblioteki „PowerPacker.library". Jeżeli używamy pierwszej metody to mogą wystąpić pewne niezgodności w rozpakowywaniu danych gdyż w przypadku pojawienia się nowej metody pakowania przez program PowerPacker nie mamy jej zaimplementowanej a ponadto zajmuje to trochę miejsca i nie jest wygodne w użyciu. Natomiast druga metoda polegająca na użyciu biblioteki PowerPacker.library zapewnia pełną kompatybilność oraz wygodę, a także dostarcza nam wielu innych możliwości, jak na przykład pakowanie danych. W tym artykule postaram się przedstawić wszystkie procedury dostępne w najnowszej wersji tej biblioteki (o numerze 35).

ppAllocCrunchInfo

cunchinfo=ppAllocCrunchInfo (efektywność, szybkość, funkcja)
D0 D0 D1 A0

Procedura przydziela wszelką potrzebną pamięć do spakowania bufora. Ta funkcja musi być wywołana przed wywołaniem procedury ppCrunchBuffer czyli procedury pakowania danych. Dla tej procedury jako dane należy przekazać efektywność oraz rozmiar bufora. Można także przekazać jej adres funkcji, która będzie wywoływana podczas pakowania danych. Funkcja może być wywoływana z dwóch powodów: aby przekazywać użytkownikowi ilość spakowanych danych lub też aby dać użytkownikowi możliwość przerwania pakowania. Do funkcji użytkownika będą przekazane trzy argumenty (argumenty są przekazywane na stosie):

lensofar - długość danych, które uległy już spakowaniu
crunlen - długość danych spakowanych
totlen - długość całkowita bufora

Jeżeli wywołujemy funkcję to musimy przechować zawartość wszystkich rejestrów. Przy wyjściu z funkcji w D0 zwracamy wartość TRUE lub FALSE. Jeżeli będzie to wartość FALSE (ujemna) to nastąpi przerwanie pakowania danych.

Wejście:

efektywność - efektywność pakowania danych:

CRUN_FAST (0) - najszybsza
CRUN_MEDICORE (1) - średnia
CRUN_GOOD (2) - dobra
CRUN_VERYGOOD (3) - bardzo dobra
CRUN_BEST (4) - najlepsza

szybkość - typ bufora jaki ma być przydzielony dla danej efektywności (przy dużym buforze pakowanie danych następuje szybciej). Podane zostały rozmiary jakie są przydzielane dla poszczególnych buforów (w kilobajtach):

SPEEDUP_BUFFSMALL (0) - od 3K (fast) do 33K (best)
SPEEDUP_BUFFMEDIUM (1) - od 5K (fast) do 65K (best)
SPEEDUP_BUFFLARGE (2) - od 196K (fast) do 256K (best)

funkcja - funkcja, która będzie wywoływana bardzo często podczas pakowania danych. Może być użyta do wyświetlania ilości spakowanych danych bądź aby umożliwić użytkownikowi przerwanie pakowania.

Wyjście:

crunchinfo - wskaźnik struktury crunchinfo, która będzie przekazana do procedury ppCrunchBuffer podczas pakowania danych (aby zwolnić przydzieloną pamięć przez ppAllocCrunchInfo należy użyć procedury ppFreeCrunchInfo). Raz przydzielona struktura "crunchinfo" może być użyta do spakowania kilku buforów danych.

---

ppCalcChecksum

suma = ppCalcCheckSum (ciąg)
D0 A0

Ta funkcja oblicza 16 bitową sumę kontrolną ciągu znaków. Suma używana jest aby sprawdzić czy hasło do odkodowywania danych jest poprawne (wyższa połowa długiego słowa zwracanego w D0 jest zerowana).

Wejście:

ciąg - wskaźnik do ciągu znaków zakończonych zerem.

Wyjście:

suma - suma kontrolna ciągu znaków.

---

ppCalcPasskey

klucz = ppCalcPasskey (hasło)
D0 A0

Procedura oblicza klucz do rozkodowania danych na podstawie podanego hasła.

Wejście:

hasło - ciąg znaków zakończony zerem.

Wyjście:

klucz - klucz odpowiadający dla zadanego hasła

---

ppCrunchBuffer

długość=ppCrunchBuffer (crunchinfo, bufor, długośćbufora)
D0 A0 A1 D0

Funkcja umożliwia spakowanie bufora danych. Należy bardzo uważać gdy się przekazuje parametry do tej funkcji. Można bardzo łatwo zawiesić system gdy się przekaże złe parametry bądź poprzez napisanie złej funkcji wywoływanej podczas pakowania danych.

Wejście:

crunchinfo - wskaźnik struktury crunchinfo zwracany przez procedurę ppAllocCrunchInfo.

bufor - wskaźnik bufora danych, który ma być spakowany

długośćbufora - długość bufora danych, który ma być spakowany

Wyjście:

długość - długość spakowanych danych. W przypadku błędu "długośćbufora" może być równa wartości PP_CRUNCHABORTED (0) bądź PP_BUFFEROVERFLOW (0).

ppDecrunchBuffer

ppDecrunchBuffer (endcrun, decrbuf, efektywność, kolor)
A0 A1 A2 D0

Funkcja umożliwia rozpakowanie danych z jednego miejsca pamięci w drugie. Adres gdzie dane będą rozpakowywane może być bliski adresowi startu spakowanych danych. Należy jednak uwzględnić pewien margines - adres pod którym dane będą rozpakowywane musi być o osiem większy niż adres spakowanych danych. Chcąc używać tej procedury musimy poznać format spakowanych danych:

1 długie słowo - identyfikator - 'PP20' lub 'PX20'

Identyfikator 'PP20' odnosi się do danych tylko spakowanych natomiast 'PX20' dla danych spakowanych i zakodowanych. W przypadku gdy wystąpił identyfikator 'PX20' to dalej następuje:

1 słowo - suma kontrolna - $ssss

Suma kontrolna służy do sprawdzenia poprawności wprowadzanego hasła.

1 długie słowo - efektywność - $eeeeeeee

X długich słów - spakowane dane - $cccccccc, $cccccccc, ...

1 długie słowo - decrunch info - $llllllkk

Decrunch info zawiera długość pliku po rozpakowaniu - jest ona przesunięta o 8 oraz 8 bitów innych wiadomości.

Najpierw należy odczytać "decrunch info" aby odnaleźć długość rozpakowanego zbioru. Następnie należy przydzielić pamięć dla danych (przesuwamy "decrunch info" w prawo o 8 bitów, a następnie dodajemy 8 bajtów na margines bezpieczeństwa aby otrzymać długość tego bloku pamięci). Jeżeli zbiór jest zakodowany należy najpierw wywołać procedurę ppDecrypt aby odkodować zbiór (przed wywołaniem procedury ppDecrunchBuffer).

A teraz krótki opis postępowania ze zbiorami spakowanymi:

- odczytać identyfikator
- jeżeli 'PX20' to odczytać sumę kontrolną (16 bitów)
- odczytać efektywność
- przydzielić "decrlen" + 8 (margines bezpieczeństwa) bajtów
- wczytać spakowane dane i "decrunch info" na początek przydzielonego bloku pamięci
- jeżeli 'PX20' to zapytać o hasło za pomocą procedury ppGetPasword i sprawdzić wynik zwracany przez procedurę z sumą kontrolną odczytaną ze zbioru
- jeżeli 'PX20' to obliczyć klucz do rozkodowywania za pomocą ppCalcPasskey
- jeżeli 'PX20' to wywołać procedurę ppDecrypt aby odkodować spakowane dane (tylko spakowane dane a nie "decrunch info")
- wywołać procedurę ppDecrunchBuffer z "endcrun" ustawionym zaraz za "decrunch info", "decrbuf" ustawionym osiem bajtów za początkiem bloku gdzie zostały załadowane dane oraz "efektywność" ustawioną na długie słowo efektywności odczytane ze spakowanych danych.

Wejście:

endcrun - wskaźnik zaraz za ostatnim bajtem spakowanych danych.

decrbuf - wskaźnik do bufora gdzie będą rozpakowywane dane (musi być przynajmniej o osiem większy niż początek spakowanych danych)

efektywność - wskaźnik do tablicy efektywności

kolor - efekt użyty przy rozpakowywaniu danych:

DECR_COL0 (0) - kolor tła
DECR_COL1 (1) - kolor liter
DECR_POINTER (2) - kolor wskaźnika
DECR_SCROLL (3) - przesuwanie ekranu na boki
DECR_NONE (4) - nic się nie dzieje

---

ppDecrypt

ppDecrypt (bufor, długość, klucz)
A0 D0 D1

Procedura odkoduje obszar pamięci za pomocą podanego klucza. Musi ona być wywoływana przed wywołaniem ppDecrunchBuffer (jeżeli zbiór był zakodowany). Jeżeli wywołujemy procedurę przed ppDecrunchBuffer musimy się upewnić czy odkodowujemy właściwy obszar. Nie wolno odkodowywać ostatniego słowa!

Wejście:

bufor - adres obszaru pamięci, który ma być rozkodowany (lokacja pamięci musi być parzysta)

długość - długość obszaru pamięci do odkodowania (musi to być wielkość podzielna przez cztery)

klucz - klucz z danego hasła zwrócony przez procedurę ppCalcPasskey

---

ppEnterPassword

wartość = ppEnterPassword (screen, bufor)
D0 A0 A1

Wymagana jest obecność biblioteki reqtools.library!!! Procedura otwiera mały requester aby zapytać użytkownika o hasło. Hasło nie będzie widziane w momencie wpisywania. Następnie, jeśli użytkownik wpisał hasło i zweryfikował je następuje powrót.

Jeśli bufor zawiera hasło pojawia się w requesterze gadżet z napisem "Last" - użytkownik może wprowadzić poprzednie hasło bez ponownego wpisywania. Od programisty zależy czy wprowadzi tę cechę do programu. Jeżeli nie chce aby taka sytuacja zaistniała należy pierwszą komórkę bufora wyzerować. Zawartość bufora nie ulega zmianie jeżeli użytkownik przerwał wpisywanie hasła. Procedura automatycznie sprawdza pr_WinodwPtr (z procesu z którego została wywołana), aby odnaleźć screen na którym ma pojawić się requester (w przypadku gdy screen podaliśmy jako zero).

Wejście:

screen - wskaźnik struktury screen bądź zero.

bufor - bufor na hasło wprowadzane przez użytkownika . musi zawierać przynajmniej 17 znaków.

Wyjście:

wartość - jeżeli użytkownik wprowadził hasło (które znajduje się w buforze) to procedura zwróci wartość nieujemną natomiast wartość ujemna pojawi się w przypadku gdy użytkownik przerwał wpisywanie hasła.

---

ppErrorMessage

komunikat = ppErrorMessage (kod błędu)
D0 D0

zwraca wskaźnik do zakończonego zerem ciągu znaków zawierającego opis błędu odpowiadający dostarczonej wartości. Aktualnie pracuje tylko z błędami zwracanymi przez ppLoadData.

Wejście:

AMIGA

kod błędu - kod błedu zwracany przez ppLoadData.

Wyjście:

komunikat - wskaźnik do komunikatu opisującego błąd o danym numerze (ciąg znaków zakończony zerem).

---

ppFreeCrunchInfo

ppFreeCrunchInfo (crunchinfo)
A0

Procedura zwalnia całą przydzieloną pamięć dla struktury crunchinfo przydzielonej przez procedurę ppAllocCrunchInfo.

Wejście:

crunchinfo - wskaźnik struktury crunchinfo zwracany przez procedurę ppAllocCrunchInfo.

---

ppGetPassword

wartość=ppGetPassword (screen, bofor, długość, sumakontrolna)
D0 A0 A1 D0 D1

Wymagana jest obecność biblioteki reqtools.library!!! Procedura otwiera mały requester aby zapytać użytkownika o hasło. Procedura wraca gdy użytkownik wpisał poprawne hasło z sumą kontrolną równą sumie kontrolnej podanej przy wejściu do procedury lub gdy nie był w stanie podać właściwego hasła (użytkownik ma na to trzy szanse). Hasło nie będzie pokazywane podczas wpisywania.

Wejście:

screen - wskaźnik struktury screen, gdzie ma się pojawić requester

bufor - bufor, w którym będzie się znajdowało właściwe hasło. Musi być przynajmniej na 17 znaków.

długość - długość bufora w znakach (powinna być zawsze 16).

suma kontrolna - suma kontrolna hasła, którego poszukujemy.

Wyjście:

wartość - w przypadku gdy użytkownik nie wprowadził właściwego hasła uzyskamy wartość ujemną. Wartość nieujemna oznacza iż poprawnie wprowadzone hasło znajduje się w buforze.

---

ppLoadData

błąd=ppLoadData (nazwa, kolor, memtype, &bufor, &len, funkcja)
D0 A0 D0 D1 A1 A2 A3

Procedura ta ładuje zbiór do przydzielonego przez nią obszaru pamięci. Zbiory spakowane za pomocą PowerPacker'a będą rozpakowane, natomiast zbiory niespakowane będą załadowane normalnie. Jeżeli zbiór nie może być otworzony, to przed kolejną próbą zostanie dodane rozszerzenie ".pp". Adres załadowanego zbioru będzie dostępny w komórce "bufor", a długość w komórce "długość". Przydzieloną pamięć należy zwalniać samodzielnie.

"Funkcja" to adres procedury wywoływanej gdy procedura ppLoadData potrzebuje hasło aby odkodować (zakodowany) zbiór. Jeżeli nie życzymy sobie kodowanych zbiorów to należy argument ustawić na -1, natomiast gdy chcemy aby procedura ppLoadData używała procedury ppGetPassword do pobiereania hasła musimy wywołać tę funkcję z argumentem "funkcja" równym zero.

Jeżeli piszemy własną procedurę obsługi pobierania hasła to musimy uwzględnić parametry przekazywane nam na stosie: "bufor" w 4(a7) oraz "sumę kontrolną" w 8(a7). Bufor ma 17 bajtów długości i nasza procedura musi zwrócić w nim poprawne hasło o sumie kontrolnej równej dostarczonej sumie kontrolnej przez procedurę ppLoadData. Procedura musi przechowywać wszystkie rejestry. W D0 zwracamy zero jeżeli hasło było poprawne lub wartość ujemną w przypadku nie podania hasła przez użytkownika.

Wejście:

nazwa - wskaźnik nazwy zbioru zakończonej zerem.

kolor - efekt użyty przy rozpakowywaniu danych:

DECR_COL0 (0) - kolor tła
DECR_COL1 (1) - kolor liter
DECR_POINTER (2) - kolor wskaźnika
DECR_SCROLL (3) - przesuwanie ekranu na boki
DECR_NONE (4) - nic się nie dzieje

memtype - typ pamięci jaka ma być przydzielona dla zbioru.

&bufor - adres zmiennej w której będzie umieszczony adres pamięci przydzielonej dla ładowanego zbioru.

&długość - adres zmiennej, która będzie zawierała długość przydzielonej pamięci.

funkcja - funkcja, która będzie wywoływana aby zapytać użytkownika o hasło w przypadku gdy zbiór będzie zakodowany.

Wyjście:

błąd - 0 jeżeli wszystko jest w porządku lub jeden z niżej wymienionych kodów.

PP_OPENERR (-1) - niemożność otworzenia zbioru

PP_READERR (-2) - błąd dysku

PP_NOMEMORY (-3) - brak pamięci aby załadować zbiór

PP_CRYPTED (-4) - zbiór jest zakodowany (gdy jako "funkcję" podamy -1)

PP_PASSERR (-5) - niepoprawne hasło

PP_EMPTYFILE (-6) - zbiór pusty (nie ma nic do załadowania)

PP_UNKNOWNPP (-7) - spakowany nieznaną wersją PowerPacker'a

---

ppWriteDataHeader

wartość=ppWriteDataHeader (handle, efektywność, kod, suma kontrolna)
D0 D0 D1 D2 D3

Funkcja jest używana do zapisu nagłówka dla zbiorów spakowanych PowerPacker'em.

Wejście:

handle - file handle otrzymany z procedury Open (dos.library) dla zbioru, do którego ma być zapisany nagłówek.

efektywność - efektywność użyta do spakowania bufora.

kod - wartość nieujemna dla danych zakodowanych.

suma kontrolna - jeżeli dane zakodowane, to jest to suma kontrolna hasła użytego do zakodowania.

Wyjście:

wartość - nieujemna jeżeli wszystko poszło dobrze.

I to już wszystko o PowerPacker.library - a więc pakujmy chłopaki!

Marcin "Duddie" Dudar

OPIS ZESTAWU PUBLIC DOMAIN PACK

AMIGA

nr 15 (dysk - marzec '92)

Witam w marcowym Public Domain Pack'u.

Na początek bardzo użyteczny i estetycznie wykonany (za co należy się Orcristowi z LSD piątka) boot-block, jest to boot z Nuke 1.4b. Oto krótki opis jego funkcji:

F1 - Memory On/Off - włączanie/wyłączanie fast'u.
F2 - Drives On/Off - włączanie/wyłączanie zewnętrznych stacji dysków.
F3 - Hard reset - zimny start systemu, łacznie z ustawieniem na nowo wszystkich wektorów systemowych (zabija virusy w pamięci).
F4 - Install - nagranie tego boot-block'u na dysk w stacji DF0.
F5 - 60 Hz (ECS) - zmiana częstotliwoci wyświetlania obrazu (do prawidłowego działania niezbędny jest odpowiedni monitor).

Czas na giercowanie, na początek - **KIM** bardzo ciekawa gra logiczna. Na planszy 15 na 15 zaznaczone są losowe pola. Celem gry jest oczyszczenie planszy. Należy pamiętać, że po kliknięciu na jakieś pole wszystkie pola dookoła zmienią swój stan. Jest to bardzo proste, ale do czasu...

Następna propozycja to **Power Wars**, niezła gra strategiczna podobna nieco do szachów, ta gra to jakby kosmiczne szachy. Walczą ze sobą dwie eskadry kosmolotów. Grę urozmaicają rozmaite rodzaje pól, teleportacje, wiele rodzajów statków, itd. Oto parametry poszczególnych statków:

	Siła (Atak)	Tarcza	Poruszanie	Wartość
Repair Ship	1	1	1	4
Fighter	2	1	2	1
Guard	1	2	2	2
Elite Fighter	3	2	3	5
Elite Guard	2	3	3	3
Powersource				
- z tarczą	4	3	1	6
- bez tarczy	5	1	2	7

Podczas starcia porównuje się siłę atakującego z tarczą atakowanego, jeżeli atakujący ma przewagę to wygrywa i na odwrót. Do siły obrony, lub ataku dodaje się punkty za kolor pola, na którym toczy się walka. I tak:

Pole białe - dodaj 2 pierwszemu z graczy.
Pole żółte - dodaj 1 pierwszemu z graczy.
Pole pomarańczowe jest neutralne.
Pole czerwone - dodaj 1 drugiemu z graczy.
Pole czarne - dodaj 2 drugiemu z graczy.

Repair Ship nie nadaje się do walki z innymi statkami, ale jako jedyny zmienia kolor pola, na którym stanie, oczywiście na korzyść jego posiadacza. Naszym strategicznym celem jest zniszczenie Powersource'a nieprzyjaciela. Dobrą taktyką jest „wykończenie Elite Fighterów" przeciwnika, ustawienie statków na dobrych pozycjach, i jeżdżenie Repair Ship'em. Później wszystko pójdzie gładko.

Teraz poważniejsze rzeczy.

Posiadacze drukarek wszelkiej maści i rodzajów, „typowi" i „nietypowi" - to dla was! Macie powód do radości bowiem skończył się kłopot z driverami. Program **PrtDrvGen** umożliwia napisanie drivera do KAŻDEJ drukarki. Do programu dołączona jest instrukcja, (niestety po angielsku, ale czekajcie na tłumaczenie w najbliższym numerze) opisująca dokładnie krok, po kroku jak samemu napisać taki driver, lub zmieniać istniejące.



Następny z użytków to **FileMaster wersja 1.1.** Bardzo dobry program z rodzaju file manager'ów takich jak, Diskmaster 1.4, 3.05 czy 2.0 oraz CLI-mate i SID. Według mnie jest najlepszym z nich. Jego podstawową zaletą jest niezawodność. Oczywiście posiada wszystkie funkcje programów tego typu, oraz wiele, wiele innych (min. wbudowany bardzo dobry edytor plików, świetny wystrój graficzny, wygodny panel sterowniczy, rozbudowaną opcję formatowania dysków).

FixDisk! Świetny program reperujący dyski. Wystarczy kliknąć na DF0: (jeżeli posiadasz zewnętrzną stację dysków to odpowiednio DF1:, DF2:, itd.), a resztę zrobi za nas FixDisk. Program będzie naprawiał zepsuty dysk, o wszystko się pytając, a więc nie ma obawy o skasowanie czegoś co chcielibyśmy uratować (tak jak to czsami bywa z DiskDoctorem). FixDisk pozwala na nieskończoną ilość prób odczytania złego sektora czy ścieżki, w nadzieji, że wreszcie się uda. Posiada także opcję wyszukiwania skasowanych plików i nagrywania ich na dysk. FixDisk jest programem, który każdy szanujący się amigant powinien posiadać w swojej biblioteczce.

Nastepne ciasteczko to **Facc**, programik zakładający tzw. Cache dyskowe. Aby przekonać się jak to działa radzę uruchomić Diskmastera, następnie Facc'a „zdirować" dysk raz, i potem jeszcze raz. Efekt gwarantowany! Niestety, aby poprawnie działać program ten potrzebuje dość dużo pamięci, można to jednak regulować gadżetami FEVER - mniej i MORE - więcej (na panelu sterowniczym).

Well! Intro Akwilonu, nowej polskiej grupy. No cóż, na ekranie bardzo ciekawy ploter (600 „dotsów" to może mało, ale nic nie było tablicowane), ładny logos i świetna muzyka. Końcową ocenę pozostawiam czytelnikom.

Na koniec pare ładnych digitalizacji w formacie IFF. Uwaga, aby zakończyć wyświetlanie obrazka, należy nacisnąć prawy klawisz myszy.

AMBER

O zasadach nabywania naszych zestawów PDP piszemy na str. 16. Poniżej przypominamy w skrócie zawartość zestawów nr 13 i 14.

Zestaw nr 13

- Super Duper 2.01
- Sanity Copy
- Noise Packer 3.00
- Ham Sharp
- Mostra
- dema graf. i muz.

Zestaw nr 14

- NUKE SADDAM 1.4
- THIEF RIPPER 2.0
- DISK MASTER 3.05
- ZIG ZAG #3 (grafika)
- dema graf. i muz.

KĄCIK POCZĄTKUJĄCEGO KODERA CZ.12

AMIGA

Screen

Dzisiaj, na prośbę naszych czytelników, rozpoczynamy cykl na temat intuition.library oraz graphics.library dotyczących komunikacji z użytkownikiem poprzez okna, menu, gadżety, etc.

Na początek zajmiemy się screen'ami czyli ekranami oraz oknami. Ekran jest to standardowy obszar biblioteki intuition jeżeli chodzi o komunikację z użytkownikiem (wyświetlanie danych, etc.). Ekran determinuje ilość kolorów jaką możemy użyć, ich rodzaj a także rozdzielczość. Każde okno, ikona, gadżet są połączone z ekranem. Gdy „ruszamy" ekranem (czy to programowo czy też za pomocą myszy) to wszystkie obiekty przydzielone do tego ekranu poruszają się razem z nim. Możemy posiadać kilka ekranów uruchomionych w tym samym czasie o różnych rozdzielczościach, kolorach, etc. na jednym ekranie monitora. Ilość ekranów jest limitowana ilością dostępnej pamięci typu ChipMem (GraphicsMem).

W Kickstarcie v1.x mamy do dyspozycji dwa typy ekranów:

*** Workbench Screen**

Jest to standardowy ekran intuition, normalnie uruchamiany w rozdzielczości Med-Res (640*256 punktów) i w czterech kolorach (dwa bitplene'y). Istnieje możliwość zmiany jego rozdzielczości i ilości kolorów (używa się tego rzadko; wyjątkowo przy zmniejszeniu liczby kolorów do dwóch - przez co oszczędza się trochę pamięci). Jeżeli piszemy małe programy, które nie wymagają specjalnych rozdzielczości lub specjalnej liczby kolorów, a ich komunikacja nie odbywa się w skomplikowany sposób za pomocą menu, to zaleca się uruchamianie ich na ekranie Workbench'a. Oszczędza to dużo pamięci potrzebnej na poszczególne bitplane'y wchodzące w skład ekranu.

*** Custom Screen**

Jeśli chcemy uzyskać ekran z określoną liczbą kolorów, w określonej rozdzielczości tudzież aby otrzymać własny obszar wyświetlania danych, powinniśmy użyć ekranu typu Custom (ekran użytkownika). Możemy określić jego rozdzielczość, ilość kolorów, przydzielić odpowiednią czcionkę, etc.

Amiga do komunikacji z poszczególnymi procedurami używa struktur czyli obszarów danych, w których przekazujemy parametry. Czasami po powrocie z procedury otrzymujemy adres innej struktury z uwzględnionymi naszymi parametrami. Jednak nie zawsze jako parametry przekazywane są struktury. Czasami jednocześnie przekazywana jest struktura i pewne parametry. Teraz poznamy objaśnienia jakie będą używane przy opisie struktur.

LONG - długie słowo
WORD - słowo
BYTE - bajt
ULONG - długie slowo bez znaku
UWORD - słowo bez znaku
UBYTE - bajt bez znaku
APTR - wskaźnik (adres do danych)
BPTR - wskaźnik w BCPL'u (adres danych podzielony przez cztery)
CHAR - obszar danych o określonej długości wypełnionych tą samą wartością
STRUCT - struktura zawarta w danej strukturze

Przy nazwach procedur będzie używane inne nazewnictwo. Np. Dos.Open(Nazwa) będzie oznaczać, iż mamy wywołać procedurę Open z biblioteki dos.library podając jako parametr etykietę Nazwa.

EKRAN

Jeżeli mamy zamiar użyć ekranu typu Custom Screen musimy zainicjować strukturę NewScreen z odpowiednimi parametrami, następnie otworzyć ekran poprzez wywołanie procedury Intuition.OpenScreen(NewScreen). Zanim jednak zajmiemy się wpisywaniem parametrów do struktury musimy podjąć decyzję dotyczące ekranu. Najpierw określamy w jakiej rozdzielczości chcemy otrzymać ekran: Low-Res (320*256 punktów) lub Med-Res (640*256 punktów). Dla każdego z nich możemy jednocześnie włączyć interlace co daje nam dwa dodatkowe tryby: Lace-Res (320*512 punktów) i Hi-Res (640*512 punktów). Oczywiście ekrany mogą używać trybu overscan (czyli bez borderów) - co może nam powiększyć trochę rozdzielczość. W trybie niskiej rozdzielczości (320 punktów w poziomie) możemy użyć od jednego do sześciu bitplene'ów, daje to odpowiednio: 2,4,8,16,32,64 kolorów i tryb HAM (4096 kolorów) włączany po ustawieniu znaczników. W trybie wysokiej rozdzielczości (640 punktów) możemy włączyć od jednego do czterech bitplane'ów co da nam odpowiednio: 2,4,8,16 kolorów. Paletę kolorów możemy załadować używając funkcji Graphics.LoadRGB (rastPort, colors, count) lub funkcji Graphics.SetRGB(viewPort, index, r, g, b). Dla danego ekranu możemy przyporządkować określoną czcionkę, która będzie wykorzystywana przy operacjach tekstowych na ekranie. Systemowa czcionka, zapisana w pamięci ROM Amigi jest zawsze dostępna i używana do wydruku danych. Czcionka ta nazywa się Topaz i istnieje w dwóch wymiarach:

8 - 80/40 znaków na linię
9 - 64/32 znaki na linię

Wymiary i pozycja ekranu nie są określone i można zdefiniować ekran na przykład na połowie wysokości, ale należy pamiętać, iż później użytkownik będzie mógł „podciągnąć" ten ekran do góry.

Na górze ekranu (tak zwany „drag bar") znajduje się nazwa ekranu. Istnieją dwa typy nazwy ekranu:

* nazwa standardowa, zdefiniowana w strukturze NewScreen

* nazwa obecna, która jest taka sama, jak nazwa aktualnie aktywnego okna (będzie to wytłumaczone w artykule traktującym o oknach).

A teraz omówimy strukturę NewScreen

struct NewScreen

00	WORD	LeftEdge,TopEdge	; lewy i górny róg ekranu - zawierają pozycje definiowanego ekranu, standardowo przyjmuje się 0,0
04	WORD	Width,Height	; szerokość i wysokość ekranu (wartości te podajemy w pikselach)
08	WORD	Depth	; głębokość ekranu czyli ilość używanych bitplene'ów (dla wysokiej rozdzielczości liczba pomiędzy 1 i 4, dla niskiej rozdzielczości liczba z zakresu 1 do 6)
0a	UBYTE	DetailPen	; rejestr koloru używany do wypisywania tekstu na ekranie
0b	UBYTE	BlockPen	; rejestr koloru używany do wypełniania obszarów na ekranie, etc.
0c	UWORD	ViewModes	; znaczniki odpowiadające za

```
                 tryb w jakim wyświetlany jest
                 obraz na ekranie; można używać
                 jednego bądź kilku, jednak niektóre
                 z nich się wykluczają, np. HIRES
                 i HAM nie mogą być w tym samym
                 czasie ustawione gdyż nie można
                 włączyć trybu Hold And Modify w
                 trybie wysokiej rozdzielczości.
0e UWORD  Type          ; typ ekranu (powinno się ustawić
                 CUSTOMSCREEN czyli ekran
                 użytkownika), można także ustawić
                 znacznik CUSTOMBITMAP jeżeli
                 chcemy dołączyć własny bitmap
                 (zdefiniowany i zainicjowany)
                 - więcej na ten temat będzie
                 wyjaśnione w kolejnych artykułach.
10 APTR  Font           ; wskaźnik zainicjowanej struktury
                 TextAttr odpowiadającej za
                 czcionkę. W przypadku podania
                 zera, aktualna czcionka zostanie
                 przypisana do tego ekranu.
14 APTR  DefaultTitle   ; wskaźnik do ciągu znaków, który
                 będzie używany jako tytuł naszego
                 ekranu (tekst musi być zakończony
                 zerem)
18 APTR  Gadgets        ; wskaźnik gadżetów przypisanych
                 do ekranu. Nie jest to używane
                 dlatego należy go wyzerować.
1c APTR  CustomBitMa    ; jeżeli chcemy podłożyć własną
                 strukturę BitMap do tego ekranu
                 powinniśmy wpisać do tej komórki
                 adres zainicjowanej struktury.
                 Należy pamiętać aby ustawić
                 znacznik CUSTOMBITMAP w
                 znacznikach Type. Jeżeli nie
                 definiujemy własnej struktury,
                 należy ustawić to pole na zero
                 a wtedy system wygeneruje
                 własną strukturę.
```

I oto cała struktura NewScreen. Jak widać ma ona długość $20 (32) bajty i taki obszar pamięci musi być dla niej zarezerwowany.

A teraz kilka słów na temat znaczników ViewModes:

HIRES	- włączenie wysokiej rozdzielczości
INTERLACE	- włączenie trybu interlace
SPRITES	- jeżeli zamierzamy używać sprite'ów należy ustawić ten znacznik
DUALPF	- włączenie trybu Dual Playfield
HAM	- włączenie trybu Hold And Modify

Aby otworzyć ekran należy zdefiniować wyżej podaną strukturę i podać ją jako parametr do funkcji Intuition.OpenScreen(NewScreen) w wyniku działania której otrzymamy wskaźnik struktury Screen i zostanie otwarty ekran bądź w przypadku niemożności otwarcia ekranu (na przykład złe dane lub brak pamięci) zostanie zwrócone zero.

Przykład:

Aby otworzyć ekran w Med-Res'ie o liczbie kolorów cztery i wymiarach 640*256 należy wykonać następującą procedurę:

```
Exec                equ   4
OldOpenLibrary      equ   -408
OpenScreen          equ   -198
CloseScreen         equ   -66
HIRES               equ   $8000
CUSTOMSCREEN        equ   $000f
        move.l Exec,a6          ; baza biblioteki
Exec
        lea   IntName(pc),a1
        jsr   OldOpenLibrary(a6); otworzenie
                                  biblioteki Intuition
        move.l d0,IntBase
; baza biblioteki
                                  Intuition
        move.l IntBase,a6
        lea   NewScreen,a0
        jsr   OpenScreen(a6)    ; otwarcie ekranu
        tst.l  d0
        beq.s  Error       ; jeżeli w d0 jest zero to
                             oznacza niemożność
                             otwarcia ekranu
        move.l d0,Screen
Maus    btst  #6,$bfe001        ; test lewego
                                  przycisku myszy
        bne.s  Maus        ; oczekiwanie na lewy przycisk
                             myszki
        move.l IntBase,a6
        move.l Screen,a0
        jsr   CloseScreen(a6)   ; zamknięcie ekranu
        moveq  #0,d0
        rts
Error   moveq  #-1,d0
        rts

NewScreen  dc.w  0,0            ; lewy górny róg ekranu
        dc.w  640,256   ; wymiary ekranu
        dc.w  2         ; ilość bitplaneów
        dc.b  0,1       ; kolory
        dc.w  HIRES     ; znaczniki ViewModes
        dc.w  CUSTOMSCREEN ; znaczniki Type
        dc.l  0         ; czcionka aktualna
        dc.l  ScreenTitle       ; nazwa ekranu
        dc.l  0         ; gadżety (brak)
        dc.l  0          ; BitMap kreowany przez
                           Intuition
; koniec struktury NewScreen
Screen    dc.l  0
IntBase   dc.l  0
ScreenTitle     dc.b  'Oto jest ekran',0
IntName   dc.b  'intuition.library',0
        END
```

Po otwarciu ekranu funkcja Intuition.OpenScreen(NewScreen) zwraca nam wskaźnik struktury Screen, który jest wykorzystywany przy wszystkich operacjach z ekranem (np. otwieranie okien, zmiana kolorów, etc.). Struktura Screen zawiera wiele ważnych informacji i dlatego należy ją poznać. Wygląda ona następująco:

```
    struct   Screen
00 APTR   NextScreen  ; wskaźnik następnej struktury
                        Screen bądź zero jeżeli ekran został
                        otwarty jako ostatni
04 APTR   FirstWindow ; wskaźnik pierwszej struktury
                        Window czyli pierwszego okna
                        otwartego na danym ekranie
08 WORD   LeftEdge,TopEdge    ; pozycja ekranu (lewy
                                górny róg)
0c WORD   Width,Height    ; wymiary ekranu
                            (szerokość razy wysokość)
10 WORD   MouseY,MouseX     ; pozycja kursora myszki
                              względem lewego górnego
                              rogu ekranu
```

AMIGA

14 WORD Flags ; użyte znaczniki
16 APTR Title ; wskaźnik tekstu wyświetlanego jako aktualna nazwa ekranu
1a APTR DefaultTitle ; wskaźnik nazwy ekranu
1e BYTE BarHeight, BarVBorder, BarHBorder, MenuVBorder, MenuHBorder
23 BYTE WBorTop, WBorLeft, WBorRight, WBorBottom, KludgeFill00 ; baity od $1e do $27 są używane wyłącznie przez system do określenia szerokości górnego paska ekranu i okna dla tego ekranu
24 APTR Font ; wskaźnik struktury TextAttr - przypisanej czcionki dla danej struktury Screen
28 STRUCT ViewPort ; struktura ViewPort opisująca tryb wyświetlania ekranu
54 STRUCT RastPort ; struktura RastPort opisująca wypełnienie ekranu
b8 STRUCT BitMap ; struktura BitMap zawierająca adresy bitplene'ów
e0 STRUCT Layer ; struktura Layer
146 APTR FirstGadget ; wskaźnik pierwszego gadżetu
14a BYTE DetailPen, BlockPen ; ustawione kolory do rysowania i wypełniania
14c WORD SaveColor00 ; kolor użyty do wykonania funkcji DisplayBeep() czyli chwilowej zmiany koloru
14e APTR BarLayer ; wskaźnik struktury Layer
152 APTR ExtData ; dodatkowe dane
156 APTR UserData ; wskaźnik danych użytkownika

A teraz procedury używane przy pracy z ekranem:

OpenScreen() - Intuition

Screen = OpenScreen (NewScreen)
D0 A0

Funkcja otwiera ekran z zadanymi parametrami w strukturze NewScreen.

Wejście:

NewScreen - struktura NewScreen

Wyjście:

Screen - struktura Screen

CloseScreen() - Intuition

CloseScreen (Screen)
A0

Procedura zamyka ekran wcześniej otwarty. Jeżeli mieliśmy jakieś okna na ekranie to musimy je zamknąć wcześniej. Późniejsze zamknięcie spowoduje kolizję systemu. Jeżeli nie istnieje więcej otwartych ekranów to Intuition automatycznie otworzy ekran Workbench Screen.

Wejście:

Screen - struktura Screen otrzymana z OpenScreen() - wskaźnik do ciągle otwartego ekranu.

MoveScreen() - Intuition

MoveScreen (Screen, DeltaX, DeltaY)
A0 D0 D1

Funkcja umożliwia przemieszczanie ekranu w górę i w dół (Kickstart 1.x) oraz na boki (KickStart 2.0).

Wejście:

Screen - wskaźnik do struktury Screen ekranu, który chcemy poruszać

DeltaX - długie słowo ze znakiem zawierające ilość pikseli o jaką ma być przesunięty ekarn w poziomie (dostępne tylko dla Kickstart 2.0). Ujemne wartości przesuwają ekran w lewo, dodatnie w prawo.

DeltaY - długie słowo ze znakiem zawierające ilość linii o jaką ma być przesunięty ekran w górę (wartości ujemne) lub w dół (wartości dodatnie).

ScreenToBack() - Intuition

Funkcja przenosi ekran poza wszelkie dostępne ekrany i zostawia go najbardziej z tyłu.

ScreenToBack (Screen)
A0

Wejście:

Screen - wskaźnik struktury Screen dla ekranu, który ma być przeniesiony do tyłu.

ScreenToFront() - Intuition

Funkcja przenosi ekran do przodu przed wszystkie ekrany.

ScreenToFront (Screen)
A0

Wejście:

Screen - wskaźnik struktury Screen dla ekranu, który ma być przeniesiony do przodu.

ShowTitle() - Intuition

ShowTitle (Screen, ShowIt)
A0 D0

Funkcja wymusza aby tytuł ekranu pojawiał się ponad lub z tyłu okien typu Backdrop. Tytuł ekranu zawsze jest pod wszystkimi normalnymi oknami.

Wejście:

Screen - Wskaźnik struktury Screen dla naszego ekranu.

ShowIt - wartość ujemna wymusza iż nazwa ekranu będzie pod wszystkimi oknami natomiast wartość zerowa - iż tytuł będzie ponad oknami typu backdrop.

OpenWorkBench() - Intuition

wynik = OpenWorkBench()
D0

Procedura próbuje otworzyć ekran WorkBench'a jeżeli jest wystarczająca ilość wolnej pamięci.

Wyjście:

wynik - zmienna, która mówi nam czy ekran Workbench'a został otworzony (lub był otworzony) - zawiera zero lub gdy nie ma możliwości otworzenia - zawiera wartość ujemną.

CloseWorkBench() - Intuition

wynik = CloseWorkBench()

Funkcja ta próbuje zamknąć ekran Workbench Screen jeżeli jest to możliwe. Jeżeli jakiś program korzysta z ekranu Workbench'a to funkcja nie zamknie tego ekranu. Zamknięcie Workbench Screen zwolni trochę pamięci i dlatego może być wykorzystywane w programach, które wymagają dużo pamięci. Należy pamiętać aby otworzyć Workbench Screen po zakończeni działania naszego programu (za pomocą OpenWorkBench()).

Wyjście:

wynik - wartość równa zero jeżeli można zamknąć Workbench lub wartość ujemna jeżeli takiej możliwości nie ma.

WBenchToBack() - Intuition

WbenchToBack()

Przenosi ekran Workbench Screen do tyłu, pod wszystkie inne ekrany.

WBenchToFront() - Intuition

WbenchToFront()

Przenosi ekran Workbench Screen do przodu, przed wszystkie inne ekrany.

SetRGB4() - Graphics

SetRGB4 (ViewPort, rejestr, czerwony, zielony, niebieski)
A0 D0 D1 D2 D3

Funkcja umożliwia zmianę jednego z kolorów ekranu na dowolny kolor wybrany z palety 4096 kolorów. Zanim użyjemy tej funkcji musimy otworzyć "graphics.library" (funkcja jest procedurą znajdującą się w tej właśnie bibliotece). Procedury tej używamy gdy chcemy zmienić ściśle określony kolor na ekranie. Jeżeli mamy większą ilość tych kolorów to lepiej używać procedury LoadRGB4().

Wejście:

ViewPort - struktura ViewPort znajdująca się w strukturze Screen. Aby odnaleźć strukturę ViewPort musimy wykonać następujące instrukcje:

```
    move.l Screen,a0
    lea   $2c(a0),a0        ; ViewPort = Screen+$2c
                              (patrz: struktura Screen)
```

rejestr - numer rejestru koloru z zakresu od 0 do 31.
czerwony - wartość składowej czerwonej (od 0 do 15)
zielony - wartość składowej zielonej (od 0 do 15)
niebieski - wartość składowej niebieskiej (od 0 do 15)
Przykład:

Poniższą procedurę możemy dołączyć do naszego poprzedniego tekstu źródłowego i po instrukcji: jsr OpenScreen(a6) dołączyć instrukcję wywołującą naszą procedurę, mianowicie: bsr ChangeCol.

```
SetRGB4   equ   -288
ChangeCol move.l Exec,a6
    lea   GfxName,a1
    jsr   OldOpenLibrary(a6)   ; otwarcie biblioteki
graphics.library
    move.l d0,a6
    move.l Screen,a0
    lea   $2c(a0),a0           ; ViewPort
    moveq #0,d0                ; kolor 00 czyli kolor tła
    moveq #$f,d1
    moveq #$0,d2
    moveq #$8,d3               ; zmiana koloru na $f08
                                 czyli fioletowy
    jsr   SetRGB4(a6)
    rts
GfxName  dc.b 'graphics.library',0
```

LoadRGB4() - Graphics

LoadRGB4 (ViewPort, paleta, ilość)
A0 A1 D0

Procedura LoadRGB4() umożliwia załadowanie na raz kilku kolorów. Procedura znajduje się w "graphics.library". Paleta kolorów musi być zdefiniowana taki sposób, że kolory to są słowa umieszczone jedno za drugim. Czyli dla palety 32 kolorowej będziemy mieli 32 słowa (64 bajty).

Wejście:

ViewPort - struktura ViewPort dla naszego ekranu
paleta - kolejno zapisane rejestry kolorów, które będą wpisywane począwszy od koloru zerowego.
ilość - ilość kolorów do przeniesienia z palety
Przykład:

Zamiast poprzedniej procedury możemy dołączyć tę do naszego programu źródłowego. Zmienia ona jednocześnie wszystkie (cztery) kolory.

```
LoadRGB4  equ  -192
ChangeCol move.l Exec,a6
    lea  GfxName,a1
    jsr  OldOpenLibrary(a6)   ; otwarcie biblioteki
                                graphics.library
    move.l d0,a6
    move.l Screen,a0
    lea  $2c(a0),a0           ; ViewPort
    lea  Paleta,a1
    moveq #4,d0               ; ilość kolorów do zmiany
    jsr  LoadRGB4(a6)         ; zmiana kolorów ekranu na
                                kolory zawarte w palecie
    rts
Paleta  dc.w $cde
    dc.w $000
    dc.w $fff
    dc.w $009
GfxName  dc.b 'graphics.library',0
```

Na dzisiaj to już wszystko. Zachęcam do własnego eksperymentowania oraz... czytania Kącika Początkującego Kodera.

Marcin "Duddie" Dudar

AMIGA

KTO PYTA NIE BŁĄDZI

Dzisiaj kolejna porcja odpowiedzi na Wasze listy!

1. *Nie wiem jak otwierać biblioteki!*
2. *Czy Copper wykonując Copperlist zawiesza działalność procesora głównego?*
3. *Komórka $bfec001 odczytuje stan klawiatury. Jak odczytać na raz dwa klawisze? Czy konieczny jest tu system?*
4. *Czy istnieją inne komórki sprawdzające stan innych rzeczy, np. myszy (jak sprawdzić prawy przycisk myszy, lewy znam)?*
5. *Jak obsługiwać dyskietkę bez pośrednictwa dos'u?*

(Nazwisko do wiadomości redakcji)

1. Na temat otwierania bibliotek poświęciliśmy artykuł w numerze grudniowym z roku 1990 na strone 19 pod tytułem "Kącik Kodera - Biblioteki". Wkradł się tam jednak błąd i nazwę biblioteki dla procedury OldOpen-Library podajemy nie w rejestrze A0 ale w rejestrze A1.
2. Nie. Copper jest procesorem równoległym i jedynie spowalnia pracę procesora głównego.
3. System nie jest potrzebny aby odczytywać stany klawiatury. Na przykład w grach i demach było by to trochę bez sensu, gdyż system bardzo spowalnia pracę programów. Odczytywanie kilku klawiszy na raz jest możliwe za pomocą właśnie komórki $bfec001. Jeśli wciśniemy jakiś wciśniemy klawisz to pojawia się tam „wartość" tego klawisza, natomiast gdy go puścimy to umieszczona zostaje „wartość" tego klawisza z wyzerowanym bitem zerowym. Należy teraz tylko kontrolować jaki klawisz został wciśnięty i zapalić dla niego odpowiedni znacznik a gdy pojawi się sygnał o puszczeniu go należy ten znacznik wyzerować.
4. Tak oczywiście. Na przykład prawy przycisk myszki to komórka $dff016 i należy przetestować bit o numerze 2. Natomiast współrzędne myszki możemy obliczać odczytując komórki $dff00a i $dff00b
5. O obsłudze dysku bez pośrednictwa dos'u pisaliśmy w poprzednim numerze "64+4" z marca 1992 roku w artykule "TrackDisk".

---

1. *Jeżeli moja Amiga 500 ma Kickstart 1.3 i takąż płytę to czy mogę rozbudować pamięć Chip do 1MB wlutowując kości do wolnych podstawek znajdujących się na płycie, czy muszę polecić tę przeróbkę fachowcowi? Czy kości do wlutowania mogą pochodzić ze zwykłego rozszerzenia RAM (512K Fast dołączanego od spodu Amigi)?*
2. *Czy w Amidze 500 Plus jest możliwość przełączenia Kickstart'u 2.0 na "goły" 1.3 (programowo lub sprzętowo)? Jaka jest orientacyjna cena tego komputera i czym, oprócz systemu operacyjnego, różni się on od zwykłej Amigi 500?*
3. *Czy istnieje implementacja Turbo Pascala 5.0 lub 6.0 (z IBM) na Amidze?*

Adam-Badam

1. Można wlutować kości pod warunkiem, że są one tego samego typu, mają ten sam czas dostępu i tę samą pojemność.
2. Można przełączyć Kickstart 2.0 na Kickstart 1.3 pod warunkiem że zakupimy podstawkę z przełącznikiem i Kickstart'em 1.3. Amiga 500 Plus kosztuje tylko niewiele drożej od zwykłej Amigi 500. Oprócz systemu operacyj - nego nowa Amiga 500 Plus ma wbudowane nowe układy (procesory równoległe) typu ECS, dające większe możliwości na przykład nowe tryby rozdzielczości, etc. Jest to coś w rodzaju Amigi 3000, tylko że z procesorem MC68000.
3. Tak! Nazywa się ten program HighSpeedPascal v1.0 i można na nim uruchamiać programy z IBM'a bez żadnych przeróbek.

---

1. *Jakie książki (w języku angielskim) opisują bibliotekę "dos.library" od strony programowania w asemblerze i w C?*
2. *Czy w "64+4" będzie drukowany opis tej biblioteki, jej procedur, funkcji i struktur?*
3. *Gdzie można dostać dyskietkowe wydanie "ROM Kernel Manual - Libraries & Devices"?*

Michał Plechowski

1. Biblioteka Dos.Library w języku angielskim została opisana w ROM Kernel Manual (innych pozycji traktujących o tej bibliotece nie znam).
2. W „64+4 & Amiga" opis tej biblioteki był już drukowany w numerach 1/91 i 2/91, w Kąciku Kodera.
3. Wydanie dyskowe "Rom Kernel Manual" można dostać - niestety - tylko na giełdach.

Marcin "Duddie" Dudar